Fifth Annual Canberra Conference and Workshops

AUSTRALIAN OPEN SYSTEMS USERS GROUP

·A·I·P·C·

HEWLETT PACKARD

digital

Sun

IBM

EASAMS

FUJITSU

AUSTRALIAN TECHNOLOGY RESOURCES

A conference on UNIX and Open Systems for the Canberra region

14th February, 1994

Tutorial - Special Event, Australian National University

15th February, 1994

Workshops, Australian National University

16th February, 1994

Conference, National Convention Centre

CONFERENCE

Wednesday 16th February, 1994 - *National Convention Centre - 9.00am*

A conference on UNIX and Open Systems for the Canberra Region brought to you by AUUG Inc. Lunch is sponsored by The Australian Information Processing Centre (AIPC). Morning refreshments are sponsored by Hewlett Packard and afternoon refreshments are sponsored by EASAMS.

The 'Free' Software Phenomenon

Dr. Kirk McKusick.

This talk by Dr. McKusick will explore the history, development, and growth of free software. He will explain where free software comes from and what the motivation is for its authors to provide their software for free. The locations of free software and an overview of what software is available will be given. The central issue of free software is its quality compared with commercial software. The quality issue naturally leads into a discussion of the cost versus benefits of using free software. The speaker will give his views on these trade-offs based on his extensive knowledge of working with free software contributed to the Computer Systems Research Group at The University of California at Berkeley. The talk will conclude with a description of where the free software movement appears to be going from here.

What's New in 4.4BSD

Dr. Kirk McKusick.

This talk by Dr. McKusick will discuss some of the changes summarised below that appear in 4.4BSD. Some of the major new facilities available in the 4.4BSD release are a new virtual memory system, the addition of ISO/OSI networking support, a new virtual filesystem interface supporting filesystem stacking, enhanced security and system management support, and the conversion to and addition of the IEEE Std1003.1 ("POSIX") facilities and many of the IEEE Std1003.2 facilities. There are numerous other changes throughout the system.

The World Wide Web - Hypermedia and the Internet

David Green, ANU.

An information explosion is now underway on computer networks around the globe. The name of this revolution is World Wide Web (WWW). The "Web" originated at CERN in Switzerland. It uses a client-server protocol that allows users to access hypermedia (= hypertext + multimedia) information from any server on the Internet. The Web's full power was realized early in 1993 with the release of freeware browsing programs, especially NCSA's program Mosaic. Perhaps the most important effect of the Web is that it makes on-line, electronic publishing a viable enterprise. This ability is already leading to the appearance of new institutions and new paradigms for research, teaching and communication.

The EIT Standard Environment

Steve Ball, Australian National University.

The configuration of Unix systems within FEIT is never stable, with system and user configuration files changing frequently. Rather than reacting to this constant change, the EIT Standard Environment (EITSE) has been designed to ease the configuration management burden for both System Administrators and users. EITSE works by defining a common configuration for all systems and users, and then allowing customisation at several levels.

Unix on steroids: antidoping control at Barcelona '92

Rob Ewin, Australian National University.

Antidoping control for the Barcelona '92 Olympic Games was performed by the Pharmacology Department of the Municipal Medical Research Institute in Barcelona. 1880 samples were processed in 15 days of competition generating about 10 gigabytes of data. Many new computing technologies were employed in Olympic antidoping control for the first time. Among these were the use of a LIMS, network connection of instruments, the Oracle data base system, and some sophisticated reporting modules developed within the laboratory.

Authorisation and Privacy in a Networked World

Lawrie Brown, Aust. Defence Force Academy.

This talk will survey the growing range of tools and techniques for providing authorisation and privacy, which are of increasing importance in a world of growing internetworking spanning many administrative domains. It will include security extensions to telnet and ftp; the IETF Common Authentication Technology; the Kerberos authorisation system; and competing secure email schemes PEM and PGP. It will attempt to place these tools in context, and indicate their availability and ease of use.

Code Navigation

Chris McDonald, University of Western Australia.

Examination of possibly large and numerous source code files is a task frequently required of programmers and undergraduate students alike. This paper discusses the design and implementation of a source code navigation tool, running under the X-window system, which attempts to address many of deficiencies of existing utilities. The tool displays source code, header files, call and de-

pendency graphs, cross-reference table and manual entries in multiple windows. Selection of syntactic elements within each window further describes or highlights that element in its semantic context.

Parallelism in UNIX RDBMS

Adam Green, Informix Australia.

What have database designers done to secure the future of high performance UNIX RDBMS technology? Where are database servers going in the foreseeable future? Technological advances are coming thick and fast in 'leap frog' manner, with each new vendor offering promising unheralded performance gains and ever diminishing costs. To this effect, current hardware advances are breathtaking. Where is the answer from the software vendors? With the advent of a new breed of UNIX RDBMS - and, as yet, not from the much vaunted objected oriented philosophy - a new leap forward has been made. The fundamental architecture of the technology to take the next step from existing UNIX RDBMS offers are 'core parallelism' and 'scalability'.

Applications of Voice / Video Multicasting

Merik Karman, Defence Service Homes.

Software exists today to use existing IP network infrastructure for voice, video and distributed whiteboard applications. This paper will demonstrate this software running over low bandwidth IP WAN links to support geographically dispersed systems administration staff. Public domain tools to achieve video conferencing are discussed along with an introduction to IP multicasting concepts.

TUTORIAL - SPECIAL EVENT

Monday 14th February, 1994 - *Australian National University - 9.00am*

This year we offer a special one day tutorial on UNIX systems internals by Dr. Kirk McKusick (Please see seperate brochure attached). Dr McKusick was formerly at the University of California at Berkeley where he was responsible for overseeing the development and release of both 4.3 BSD and 4.4 BSD. This tutorial will run from 9.00 am till 5.00 pm. Morning and afternoon refreshments will be provided. The tutorial will be held on the ANU campus, with easy access to various luncheon spots. You can attend the tutorial without attending the conference or workshops. The size of the tutorial is limited. Spaces will be allocated on a first come basis. We reserve the right to cancel the tutorial (if we cancel a refund will be arranged). For further information contact John Barlow on 2490747(fax) or 2492930(work) or John.Barlow@anu.edu.au(email). Registration for this tutorial is through the attached registration form.

WORKSHOPS

Tuesday 15th February, 1994 - *Australian National University - 9.00am*

This year we offer nine workshops. These workshops are a quick and enjoyable introduction to various technical aspects of UNIX systems. In general both beginners and experienced users are catered for. The first session for workshops is from 9 am till 12 noon and the second session is from 1.30 pm till 4.30pm. Morning and afternoon refreshments will be provided. The workshops will be held on the ANU campus, with easy access to various luncheon spots. You can attend the workshops without attending the conference. The size of each workshop is limited. Spaces will be allocated on a first come basis. We reserve the right to cancel any workshop (if we cancel any workshop a refund will be arranged). For further information contact John Barlow on 2490747(fax) or 2492930(work) or John.Barlow@anu.edu.au(email). Registration for these workshops is through the attached registration form.

W1 - Network Management (½ day)

Mark Turner, AARNet.

A case study will highlight, with emphasis on the UNIX platform, the strategies and tools available, and discuss the integration of public domain software and commercial turn-key NMS packages in a step towards the ideal management solution. Topics discussed will include: status monitoring; data collection for the purposes of planning and troubleshooting; traffic analysis; Management Information Bases (MIBs) and the Simple Network Management Protocol (SNMP); assorted UNIX tools.

W2 - Introduction to Internetworking (½ day)

Peter Elford, Cisco.

This tutorial is targeted at those that need or want to understand why network protocols exist and how they allow networks of networks, or internetworks, to be constructed. The basic concepts of a networked system will be presented, and compared to the ISO Open Systems Interconnection model. Detailed examples using TCP/IP will be used to show how these concepts can be applied. A review of the AppleTalk and Novell protocols will also be included.

W3 - Solaris 2.x Migration (½ day)

Merik Karman, Defence Service Homes.

This workshop will provide an overview of the Solaris environment and provide practical advice on the migration from SunOS to Solaris 2.x. The tutorial will be based on practical experience gained from using Solaris 2.x in production for more than 12 months. Topics covered will include "Whats New", "Whats Missing", transition resource material, installation and systems administration. Bring a blank 1/4" or 8mm tape to obtain useful software.

W4 - Security (½ day)

Jeremy Bishop, Department of Defence.

Is the security of your Unix computer system adequate? This workshop will cover security for both standalone and networked systems. Freely

available security software will be discussed, including system security checkers, password crackers, integrity checkers, improved authentication methods, and setting up a firewall system. Bring a 1/4" cartridge or Exabyte tape for your copy of this software.

W5 - Everyday UNIX utilities (½ day)

Ivan Angus, Australian National University.

UNIX contains a rich set of tools for the manipulation of data which can be applied to such diverse tasks as system administration, file reformatting and creating custom utilities. Such utilities as grep, sed and awk will be discussed and shell script programming will be illustrated.

W6 - The X Window System (½ day)

Bob Dynes, Labtam Australia.

An introduction to X11 Windows for beginner users. This workshop will explore basic concepts of X11, window managers and fundamental X11 user programs such as xterm. Various X11 applications and X Terminal configuration will be demonstrated. This is a hands on workshop.

W7 - Perl - the Programmer's Swiss Army Chainsaw (½ day)

Andy Linton, AARNet.

Perl is an interpreted language optimized for scanning arbitrary text files, extracting information from those text files, and printing reports based on that information. It's also a good language for many system management tasks. The language is intended to be practical rather than beautiful. It combines some of the best features of C, sed, awk, and sh, so people familiar with those languages should have little difficulty with it. Perl is defined by its author Larry Wall as meaning either 'Practical Extraction and Report Language' or 'Pathologically Eclectic Rubbish Lister'. After this workshop, you will be able to decide which.

W8 - The Tcl Language and the Tk Toolkit (full day)

Micheal Kearney, CSIRO DIT and Steve Ball, ANU.

Tcl and Tk offer a new approach to constructing X applications that is both much simpler than alternative approaches and also much more powerful. Tcl is a simple shell-like scripting language whose interpreter is implemented as a C library package. Tk is a windowing toolkit that includes a Motif-like widget set. This tutorial will provide hand-on experience in how to (a) write Tcl scripts, (b) use Tk's features to write Tcl scripts that generate user interfaces, (c) to extend the base features of Tcl with new commands implemented in C, (d) demonstrations of some Tcl/Tk extensions. Attendees should be familiar with a modern block structured progamming language, and with development in the Unix environment. Understanding of window systems concepts (in particular X windows) would be useful, but not essential.

W9 - Sendmail 8.6.4 (½ day)

John Barlow, Australian National University.

This tutorial will look at the installation and configuration of sendmail 8.6.4. Included will be information applicable to configuring most releases of sendmail and a run-down of some of the problems you can encounter. Some useful tools will also be discussed. Bring along a blank 1/4 inch or 8mm cartridge to get a copy of the software.

Registration

The attached registration form must be used for the conference, workshops and tutorial. Extra copies of the registration form are available from John Barlow on (06) 2492930, FAX (06) 2490747 or E-Mail auug94@canb.auug.org.au. Early Bird registrations must be received by 4th February 1994. Refunds will only be given under exceptional circumstances.

		Late	Early Bird
Tutorial	AUUG Members	$120	$100
	Non Members	$150	$130
Workshops - Half Day	AUUG Members	$60	$50
	Non Members	$70	$60
Workshops Full Day	AUUG Members	$120	$100
	Non Members	$140	$120
Conference	AUUG Members	$70	$50
	Non Members	$80	$60

PRINTED BY UNION OFFSET CO. PTY LTD, CANBERRA

ALFORD	Gavin	Australian Bureau of Statistics
ARMSTRONG	Tim	Australian National University
ARROWSMITH	Brad	Department of Defence
ATHERTON	Peter	Australian Taxation Office
AUER	Karl	Australian National University
BALDWIN	David	Australian National University
BALL	Steve	Australian National University
BANERJIE	Arijit	Department of Veterans Affairs
BARLOW	John	Australian National University
BARRATT	Andre	Australian National University
BEENS	Helen	CSIRO
BISHOP	Jeremy	Department of Defence
BLUHDORN	John	Australian Taxation Office
BODDY	Mark	
BOSTON	Tony	Environmental Resources Information Network
BRADY	Charlie	Telectronics Pacing Systems
BRENT	Erin	Australian National University
BRITTAIN	Chris	Department of Social Security
BROWN	Lawrie	Australian Defence Force Academy
BUCKLEY	Robin	Environmental Resources Information Network
BURHOP	Winston	Department of Primary Industries & Energy
CLACHER	Chris	CRA Exploration
CLARKE	David	Australian National University
COGSWEL	Andrew	Defence Service Homes
COHEN	Pam	Australian National University
COVINGTON	Barbara	Australian Information Processing Centre
CRIBB	Grant	Australian Electoral Commision
CROSSLEY	David	Environmental Resources Information Network
CZEZOWSKI	Adam	Australian Defence Force Academy
DANQING	Zhang	Australian Defence Force Academy
DAVIE	Peter	ITI Canberra
DAVY	Robert	Australian National University

DEBAECKER	Francois	Australian Institute of Criminology
EVERSONS	Peter	National Resource Information Centre
EWIN	Robert	Australian National University
FARMER	Wayne	Murray-Darling Bassin Commission
FARRUGIA	Greg	Australian Defence Force Academy
FORRESTER	Bob	CSIRO
GANS	Martin	ACT Government Computing Service
GRAY	Alan	InText CP Software Group
GREEN	David	Australian National University
GREEN	Adam	Informix Australia
HAMLYN	Jeremy	Australian Defence Force Academy
HINTON	Debra	Australian National University
HOFFMAN	Gaby	Australian National University
HOFMAN	Robbert	Australian National University
HOLT	David	Environmental Resources Information Network
HOWARD	Jeff	Australian Defence Force Academy
IRVING	Colin	DSTO
JENKIN	Steve	CSIRO
JOBSON	David	Australian National University
JONES	Jill	Australian National University
KARMAN	Merik	Defence Service Homes
KAYE	Tricia	Environmental Resources Information Network
KEARNEY	Michael	CSIRO DIT
LAYTON	Charles	Department of Defence
LEONARD	Raymond	Department of Defence
LEUNG	Sek Kit	InText CP Software Group
LEWIS	Ian	DSTO
LIGHTFOOT	Michael	SysIX
LIM	Mathew	Australian National University
McDONALD	Chris	University of Westerm Australia
MacKENZIE	Hugh	Gnosys Pty Ltd
McKUSICK	Kirk	

MAKIN	Carl	Department of Human Services & Health
MALBON	Rodney	Australian Electoral Commision
MARASSOVICH	Chris	Defence Service Homes
MAYO	Kevin	Sun Microsystems
MONFRIES	Wayne	Australian Taxation Office
MORRIS	Trevor	Australian Bureau of Criminal Intelligence
MUIR	Tracey	National Resource Information Centre
MURDOCH	Mark	Defence Service Homes
OLSEN	Michael	Australian Taxation Office
PALMER	Jack	Australian Seismological Centre
PEYTON	Andy	CSC Australia
PRICE	Greg	Department of Defence
PRIMMAR	Paul	Department of Defence
RATHINAM	Kannan	EASAMS
ROTHWELL	Stephen	NEC Information Systems Australia
RUSSELL	Crispin	Australian Defence Force Academy
SCARMAN	Andrew	Environmental Resources Information Network
SEARLE	Brendon	Department of Defence
THOMSON	David	DSTO
TIN	Shiu	Australian National University
TOOMEY	Warren	Australian Defence Force Academy
TRITSCHLER	Adrian	Australian Defence Force Academy
TURNER	Glen	Australian Bureau of Statistics
TYLER	Anne	Department of Primary Industries & Energy
UNG	Wen	Australian Defence Force Academy
VEAR	Eric	ACT Government Computing Service
VENNONEN	Ari	Australian Taxation Office
WALKER	Geoff	Australian Taxation Office
WHIGHT	Peter	Defence Service Homes
WILLIAMS	Chris	Department of Defence
WILLS	Jo-Anne	CSIRO

5th Annual Canberra Conference & Workshops

What's New in 4.4BSD

Dr. Marshall Kirk McKusick

1. Introduction

The major new facilities available in the 4.4BSD release are a new virtual memory system, the addition of ISO/OSI networking support, a new virtual filesystem interface supporting filesystem stacking, a freely redistributable implementation of NFS, a log-structured filesystem, enhancement of the local filesystems to support files and filesystems that are up to 2^63 bytes in size, enhanced security and system management support, and the conversion to and addition of the IEEE Std1003.1 (''POSIX'') facilities and many of the IEEE Std1003.2 facilities. In addition, many new utilities and additions have been made to the C-library. The kernel sources have been reorganized to collect all machine-dependent files for each architecture under one directory, and most of the machine-independent code is now free of code conditional on specific machines. The user structure and process structure have been reorganized to eliminate the statically-mapped user structure and to make most of the process resources shareable by multiple processes. The system and include files have been converted to be compatible with ANSI C, including function prototypes for most of the exported functions. There are numerous other changes throughout the system.

2. Changes in the Kernel

This release includes several important structural kernel changes. The kernel uses a new internal system call convention; the use of global (''u-dot'') variables for parameters and error returns has been eliminated, and interrupted system calls no longer abort using non-local goto's (longjmp's). A new sleep interface separates signal handling from scheduling priority, returning characteristic errors to abort or restart the current system call. This sleep call also passes a string describing the process state, which is used by the ps(1) program. The old sleep interface can be used only for non-interruptible sleeps.

Many data structures that were previously statically allocated are now allocated dynamically. These structures include mount entries, file entries, user open file descriptors, the process entries, the vnode table, the name cache, and the quota structures.

Both the HP300 and SPARC ports feature the ability to run binaries built for the native operating system (HP-UX or SunOS) by emulating their system calls. Though this native operating system compatibility was provided by the developers as needed for their purposes and is by no means complete, it is complete enough to run several non-trivial applications including those that require HP-UX or SunOS shared libraries. For example, the vendor supplied X11 server and windowing environment can be used on both the HP300 and SPARC.

2.1. Virtual memory changes

The new virtual memory implementation is derived from the MACH operating system developed at Carnegie-Mellon, and was ported to the BSD kernel at the University of Utah. The MACH virtual memory system call interface has been replaced with the ''mmap''-based interface described in the ''Berkeley Software Architecture Manual''. The interface is similar to the interfaces shipped by several commercial vendors such as Sun, USL, and Convex Computer Corp. The integration of the new virtual memory is functionally complete, but, like most MACH-based virtual memory systems, still has serious performance problems under heavy memory load.

2.2. Networking additions and changes

The ISO/OSI Networking consists of a kernel implementation of transport class 4 (TP-4), connectionless networking protocol (CLNP), and 802.3-based link-level support (hardware-compatible with Ethernet*). We also include support for ISO Connection-Oriented Network Service, X.25, TP-0. The session and presentation layers are provided outside the kernel by the ISO development environment (ISODE). Included in this development environment are file transfer and management (FTAM), virtual terminals (VT), a directory services implementation (X.500), and miscellaneous other utilities.

Several important enhancements have been added to the TCP/IP protocols including TCP header prediction and serial line IP (SLIP) with header compression. The routing implementation has been completely rewritten to use a hierarchical routing tree with a mask per route to support the arbitrary levels of routing found in the ISO protocols. The routing table also stores and caches route characteristics to speed the adaptation of the throughput and congestion avoidance algorithms.

2.3. Additions and changes to filesystems

The 4.4BSD distribution contains most of the interfaces specified in the IEEE Std1003.1 system interface standard. Filesystem additions include IEEE Std1003.1 FIFOs, byte-range file locking, and saved user and group identifiers.

A new virtual filesystem interface has been added to the kernel to support multiple filesystems. In comparison with other interfaces, the Berkeley interface has been structured for more efficient support of filesystems that maintain state (such as the local filesystem). The interface has been extended with support for stackable filesystems done at UCLA. These extensions allow for filesystems to be layered on top of each other and allow new vnode operations to be added without requiring changes to existing filesystem implementations. For example, the umap filesystem is used to mount a sub-tree of an existing filesystem that uses a different set of uids and gids than the local system. Such a filesystem could be mounted from a remote site via NFS or it could be a filesystem on removable media brought from some foreign location that uses a different password file.

In addition to the local ''fast filesystem'', we have added an implementation of the network filesystem (NFS) that fully interoperates with the NFS shipped by Sun and its licensees. Because our NFS implementation was implemented using only the publicly available NFS specification, it does not require a license from Sun to use in source or binary form. By default it runs over UDP to be compatible with Sun's implementation. However, it can be configured on a per-mount basis to run over TCP. Using TCP allows it to be used quickly and efficiently through gateways and over long-haul networks. Using an extended protocol, it supports Leases to allow a limited callback mechanism that greatly reduces the network traffic necessary to maintain cache consistency between the server and its clients.

A new log-structured filesystem has been added that provides near disk-speed output and fast crash recovery. It is still experimental in the 4.4BSD release, so we do not recommend it for production use. We have also added a memory-based filesystem that runs in pageable memory, allowing large temporary filesystems without requiring dedicated physical memory.

The local ''fast filesystem'' has been enhanced to do clustering which allows large pieces of files to be allocated contiguously resulting in near doubling of filesystem throughput. The filesystem interface has been extended to allow files and filesystems to grow to 2^63 bytes in size. The quota system has been rewritten to support both user and group quotas (simultaneously if desired). Quota expiration is based on time rather than the previous metric of number of logins over quota. This change makes quotas more useful on fileservers onto which users seldom login.

The system security has been greatly enhanced by the addition of additional file flags that permit a file to be marked as immutable or append only. Once set, these flags can only be cleared by the super-user when the system is running single user. To protect against indiscriminate reading or writing of kernel memory, all writing and most reading of kernel data structures must be done using a new ''sysctl'' interface. The information to be access is described through an extensible ''Management

*Ethernet is a trademark of the Xerox Corporation.

Information Base'' (MIB).

2.4. POSIX terminal driver changes

The biggest area of change is a new terminal driver. The terminal driver is similar to the System V terminal driver with the addition of the necessary extensions to get the functionality previously available in the 4.3BSD terminal driver. 4.4BSD also adds the IEEE Std1003.1 job control interface, which is similar to the 4.3BSD job control interface, but adds a security model that was missing in the 4.3BSD job control implementation. A new system call, *setsid*, creates a job-control session consisting of a single process group with one member, the caller, that becomes a session leader. Only a session leader may acquire a controlling terminal. This is done explicitly via a TIOCSCTTY *ioctl* call, not implicitly by an *open* call. The call fails if the terminal is in use.

For backward compatibility, both the old *ioctl* calls and old options to *stty* are emulated.

3. Changes to the utilities

There are several new tools and utilities included in this release. A new version of ''make'' allows much-simplified makefiles for the system software and allows compilation for multiple architectures from the same source tree (which may be mounted read-only). Notable additions to the libraries include functions to traverse a filesystem hierarchy, database interfaces to btree and hashing functions, a new, fast implementation of stdio and a radix sort function. The additions to the utility suite include greatly enhanced versions of programs that display system status information, implementations of various traditional tools described in the IEEE Std1003.2 standard, and many others.

We have been tracking the IEEE Std1003.2 shell and utility work and have included prototypes of many of the proposed utilities. Most of the traditional utilities have been replaced with implementations conformant to the POSIX standards. Almost the entire manual suite has been rewritten to reflect the POSIX defined interfaces. In rewriting this software, we have generally been rewarded with significant performance improvements. Most of the libraries and header files have been converted to be compliant with ANSI C. The system libraries and utilities all compile with either ANSI or traditional C.

The Kerberos (version 4) authentication software has been integrated into much of the system (including NFS) to provide the first real network authentication on BSD.

The *find* utility has two new options that are important to be aware of if you intend to use NFS. The ''fstype'' and ''prune'' options can be used together to prevent find from crossing NFS mount points.

3.1. Additions and changes to the libraries

The *curses* library has been largely rewritten. Important additional features include support for scrolling and *termios*.

An application front-end editing library, named libedit, has been added to the system.

A superset implementation of the SunOS kernel memory interface library, *libkvm*, has been integrated into the system.

Nearly the entire C-library has been rewritten. Some highlights of the changes to the 4.4BSD C-library:

- The newly added *fts* functions will do either physical or logical traversal of a file hierarchy as well as handle essentially infinite depth filesystems and filesystems with cycles. All the utilities in 4.4BSD that traverse file hierarchies have been converted to use *fts*. The conversion has always resulted in a significant performance gain, often of four or five to one in system time.
- The newly added *dbopen* functions are intended to be a family of database access methods. Currently, they consist of *hash*, an extensible, dynamic hashing scheme, *btree*, a sorted, balanced tree structure (B+tree's), and *recno*, a flat-file interface for fixed or variable length records referenced by logical record number. Each of the access methods stores associated key/data pairs and uses the same record oriented interface for access.

- The *qsort* function has been rewritten for additional performance. In addition, three new types of sorting functions, *heapsort*, *mergesort*, and *radixsort* have been added to the system. The *mergesort* function is optimized for data with pre-existing order, in which case it usually significantly outperforms *qsort*. The *radixsort* functions are variants of most-significant-byte radix sorting. They take time linear to the number of bytes to be sorted, usually significantly outperforming *qsort* on data that can be sorted in this fashion. An implementation of the POSIX 1003.2 standard *sort* based on *radixsort* is included in 4.4BSD.
- The floating point support in the C-library has been replaced and is now accurate.
- The C functions specified by both ANSI C, POSIX 1003.1 and 1003.2 are now part of the C-library. This includes support for file name matching, shell globbing and both basic and extended regular expressions.
- ANSI C multibyte and wide character support has been integrated. The rune functionality from the Bell Labs' Plan 9 system is provided as well.
- The *termcap* functions have been generalized and replaced with a general purpose interface named *getcap*.
- The *stdio* routines have been replaced, and are usually much faster. In addition, the *funopen* interface permits applications to provide their own I/O stream function support.

4. The End of BSD from Berkeley

The 4.4BSD distribution is the final release that will be done by the Computer Systems Research Group (CSRG). For the following three reasons, the CSRG clearly could not continue in its present form.

Funding had become increasingly time-consuming and difficult. We were spending more and more of our time obtaining funding, time that we would have preferred to spend working on BSD. As many of you are intimately aware, computer corporations are actively seeking ways to reduce discretionary outlays. Also, as UNIX vendors have developed their own research groups, the work of the CSRG became less necessary to them. Finally, making BSD freely redistributable resulted in fewer distributions sold, as other organizations sold our releases for less money.

Support within the University of California declined as BSD became less widely used internally. Victims of our own success, many of the features once found only in BSD are now available from every vendor.

The system has become too large and complex for a group of four to architect and maintain. In the last few years it became obvious to us that we had to expand the size of our group if we wanted to continue developing and distributing a complete UNIX system. Expansion was impossible given the external funding environment and the space constraints imposed by the university.

BSD has always been a community effort, and, as a community effort, does not rely on a small group of people in Berkeley to keep it going. BSD will not go away, but will live on through the free software and commercial efforts of many people. Many of these efforts are already well underway. In the free software arena there are the groups supporting 386BSD, NetBSD, and FreeBSD. On the commercial side is Berkeley Software Design, Inc. Those of us at the CSRG thank you for your support over the years, your funding, and, of course, the software you've contributed to make the BSD system what it is today!

Code Navigation

Chris McDonald

chris@cs.uwa.edu.au
Programming, Systems and Algorithms Laboratory,
Department of Computer Science, The University of Western Australia,
Crawley, Western Australia, 6009.

ABSTRACT

Examination of possibly large and numerous source code files is a task frequently required of programmers and students alike. A number of Unix utilities exist which attempt to summarize or better present source code, but none work from attributed syntax trees, follow preprocessor conditional compilation or provide contextual information beyond printing lines from the source code. This paper discusses the design and implementation of a source code navigation tool, running under the X-window system, which attempts to address many of deficiencies of existing utilities. The tool displays source code, header files, call and dependency graphs, cross-reference table and manual entries in multiple windows. Selection of syntactic elements within each window further describes or highlights that element in its semantic context.

1. Introduction

Examination of possibly large and numerous source code files belonging to a single project is a task frequently required of programmers and students alike. Rapidly emerging technologies such as internetworking ensure that there will forever be more examples of good, and bad, coding practices than can be comprehended by any individual. However, the realization of software engineering ideals, such as clear and entertaining documentation, have not kept pace and the task of reading volumes of source code to understand its workings is often, still, necessary. When the source code under scrutiny has been written by someone else, or just retrieved from paper tape, its global structure and components are not always readily apparent. Cross-reference generators and pretty-printers provide much needed illumination. Greater satisifaction arises from reading either seminal source code or a package offering suitable code-reuse. A clear understanding of source code is often a requirement of our undergraduate students at The University of Western Australia. Students are frequently required to understand code presented in a number of source files, either because of its inherent pedagogical value or to enhance or correct it. Examples occur in our *Programming Language Implementation* course, in which students extend a 1200 line compiler for a Pascal-like language, and in our *Computer Networking* course, in which seminal network protocol implementations are examined.

Despite the overwhelming acceptance of window-based programming environments, few satisfactory tools exist within these environments to assist the task of *code navigation.* In this context, code navigation is loosely defined as the process of traversing and enquiring about source code (ideally within an assisting environment, or tool) to learn about the code's structure and contents. Although code navigation is considered a necessary component in the overall task of programming, its definition here does not extend to include either editing or debugging.

2. Motivation and Design

The work described in this paper was motivated by that of Hutchinson *etal* [HTJ91] which reports on a Macintosh Hypercard stack for the source code associated with the XINU operating system project. In this work, each of the near 100 source files of the XINU project, which in total contains over 5100 lines of C and 600 lines of 68000 assembler, is managed as a single Hypercard card. Each card presents its source code on a scrollable panel and provides buttons to navigate to the next or previous card (source file). The source code is closely indexed with reference to the XINU textbook [CM89], so much so that as the source code is scrolled, another numeric panel updates to reflect the page number of the text on which the source code is described.

Many of the C function and variable identifiers on the source code panels are *active regions* (in Hypercard parlance). Their selection with the mouse introduces a Macintosh dialogue box which further reports the identifier's place of definition (source file name and textbook page number). Selection of a button on the new dialogue box changes the source code focus by displaying the (another) Hypercard card presenting the location of the identifier's definition. This process may be repeated, with the "pushing" and "popping" of cards, as identifiers and their use are tracked through the XINU files and textbook.

The XINU source code navigation environment has been used successfully by its authors in their undergraduate operating system course and they report an increased interest by students as they wade through the plethora of files and functions in the course. Unfortunately, the Hypercard stack has been designed and constructed as a static environment – while it successful permits navigation through the XINU source code, it is constrained to those files. The Hypercard stack has been developed "by hand", only the identifiers considered as significant and indexed by the authors are "active" and can be tracked. The project begs the question as to how another source code collection could be indexed, for example if the MINIX project was selected for their operating system course. The authors would have clearly used source code (or even textual) cross-referencing tools to build the necessary index, and Douglas Comer is acknowledged as having helped with this task. It is unfortunate that the idea was not pushed further.

There are, of course, a number of utilities in the public domain which perform the tasks of cross-referencing, summarising and presenting bodies of source code, particularly in the C programming language. Two such utilties, often cited on USENET, are `cflow` and `calls`. However, none appears to address more than a handful of issues, nor attempts to position itself within the historical UNIX toolkit framework of producing its output for filtering by another program. Given the overwhelming acceptance of windowing interfaces to the UNIX operating system, this is not surprising. What is a little more difficult to understand is that few of the publically available utilities perform their task in a correct manner and can often be confused by non-standard coding practices. This situation arises, in general, because none of the utilities correctly preprocess or parse their source code, as must a compiler.

For example, the popular source code formatter, `indent`, does not parse its input and simply builds a token stack (rather than a syntax tree) which is maintained until the stack contents need reformatting. For this reason, `indent` must be informed with command line arguments which tokens are user-defined types. `hat_and_coat`, the Header Analysis Tool and Common Object Analysis Tool (!), attempts to locate all `#defines`, **`typedefs`**, **`enums`**, **`structs`**, and **`unions`** from header (and occasionally C) files and produces a topological graph reporting the definitions. `hat_and_coat` does not correctly recognize C syntax, and attempts to parse files before presenting them to the preprocessor. The result is that preprocessor macros are reported as function calls and

conditionally compiled code may be incorrectly cross-referenced. Similarly, `unifdef`, which filters lines of a C program not in effect due to conditional preprocessing, does not fully recognize conditional expressions such as `#if defined(X) || defined(Y)`.

The previous paragraph is not intended to be a criticism of all earlier work. Instead, it attempts to highlight the difficulties associated with correctly cross-referencing and presenting source code and, hopefully, provides some motivation that the difficult task needs addressing.

Like the XINU Hypercard system, the unimaginatively named `xc` program has been designed to enable navigation through source code. However, the primary design goal has been to enable any collection of code to be traversed, with information being gleaned from the code and the environment in which it would normally be compiled and executed. The initial implementation of `xc` has been attempted under the SunOS 4.1.3 operating system, and uses the XView toolkit and Xlib libraries under X11R5. All times reported in this paper were taken from a SPARC-1+ workstation with 52Mb of memory. `xc` is far from complete, though its present implementation demonstrates that many early design goals can be met.

3. Execution of the `xc` tool

The `xc` program commences execution by first considering its command line options and then reading or parsing the contents of files whose names are passed as arguments (such files are assumed to contain C source code or be object or archive files, depending on their filename extensions). `xc` is intended to be a "drop-in" replacement for a command line which would typically be passed to a C compiler to compile and link many source files and libraries (actually counter to the methods dictated by the use of *make*). In this role, `xc` supports many of the expected preprocessor, compiler and linker options, silently ignoring all others (`-O` is sarcastically ignored). For example, a command line to invoke `xc` on a single file of its own source code is

```
xc -DSUNOS -I/usr/local/X11R5/include -L/usr/local/openwin3.0/lib \
           xc.c -lxview -lolgx -lX11
```

Here the `-D` and `-I` options request the same actions as those performed by the C preprocessor, and the `-L` and `-l` options, the actions of the linker.

Having built symbol tables and indexes of the source files, `xc` displays the first file in a scrollable canvas, with additional buttons to control its presentation. Each canvas also responds to commands from the keyboard – single character sequences control absolute, forward, backward, left and right motion and lines within the source code may be marked and revisited (character sequences unashamedly mimic those of `vi` and `less`).

Program keywords are presented in a emboldened font, strings and comments in an oblique font. All fonts are fixed width, which enables program indentation to be easily preserved. This is in sharp contrast to the impressive work of Baecker and Marcus [BM90] in which they report their extensive study of font attributes and sizes which promote clarity. Moreover, Baecker and Marcus argue that emboldened fonts should *not* be used for program keywords. Their research is now a little dated, in that it only addresses the presentation of programs on paper, and in particular addresses pagination and titling in detail. Scope exists for their work to be extended to window-based environments, though the *typographic* presentation of source code has not been a significant concern of `xc`.

Unlike the Hypercard environment, all text in `xc`'s source code canvas is active. Where possible, selection of the text results in that selection being highlighted and described in its context. For example, selection of an identifier (with the left mouse button), such as `argv`, results in `argv` being highlighted in reverse video, and a description of that identifier being reported in the frame's footer. Each identifier is only highlighted within the implied lexical scope; an identifier also named `argv` in a function other than, say, `main` is not highlighted. At present, little is reported about `argv` other than that is an identifier. Future work will report the data type of `argv` and announce it as a formal parameter of `main`. Selection of a C keyword provides a short explanation of the keyword, and basetypes, such as **`long`**, are reported as being (here) 32 bits long.

Returning to our example, if the selected identifier is `strcmp`, it is highlighted (everywhere, as it is an external function). `strcmp` is further described as an external function, that it is defined in the library `libc.a` and that there is a UNIX manual describing it in section 3 of the manual (the section describing user-level library functions).

Selection of an identifier, such as `strcmp`, with the right mouse button results in a popup menu being displayed on which relevant manual sections are announced. In the case of `strcmp`, only section 3 will be available (other manual sections are inactive); if the identifier were `exit`, sections 1, 2 and 3 could be selected.

Selection of an active manual section from the popup menu results in a new window appearing which presents the selected UNIX manual entry. This window is similar to the initial source code window – the manual information appears on a scrollable panel with font changes delimiting sections of the text. `xc` actually reads the relevant `catman` entry of the required manual (itself, `nroff` output), or the `man` output if the `catman` directory is not writable. The manual's title and subsection headings appear as emboldened text, emphasized manual contents as reverse video. In many respects, this echoes the appearance of the standard `xman` utility. Like source code windows, all text on manual windows is active – selection of a word (akin to an identifier) results in an (often uninformed) description of that word. If the word has a corresponding manual entry, then that manual may also be displayed, a useful feature when `SEE ALSO` subsections are followed. Again, this echoes the activities of the freely available `Tkman` tool.

Other active text regions of interest are the C preprocessor `#include` directives. Selection of a `#include` directive from either a source or manual entry window creates a new source window to display the requested header file. References to both absolute and relative filenames are supported; absolute references are located in directories provided with `-I` command line arguments and then the standard directory (`/usr/include`, unless `-Y` is presented), relative filenames are first sought in the directory of their "parent" file and then as for absolute filenames.

Each source code window also provides two additional buttons. Selection of the first, labeled `Include tree`, creates a new window on which all header files, included by the current file with a `#include` directive, are presented in a horizontal tree structure. The current filename appears to the very left (the *root* of the tree), files it `#includes` are presented to its right with arrows connecting the filenames. Files subsequently `#included` are presented further right, and so on. Backward pointing arrows, presented as dashed lines, represent recursive (or circular) file inclusion which is typically (hopefully) guarded against by conditional compilation directives. The current representation is rather crude, little attempt is made to present a clear representation. Further work, such as that reported by Vaucher and Bloesch [VAU80][BLO93], needs to be incorporated. All text fields (filenames) on this `#include` window are also active – selection of a filename creates a new

window presenting the requested header file.

The other button on each source window, labeled `Reformat`, displays a selection panel from which about twenty attributes may be modified to control the appearance of the source file window. Most attributes are boolean, and are selected using non-exclusive buttons. Numeric attributes are controlled with integral sliders. The reformatting of the source code is, of course, not performed by `xc` itself. Instead, the current contents of the source window is presented as standard input to the `indent` utility; its standard output is redirected to a file and the contents of this file are displayed on the original source window. The attribute buttons and sliders control the command line arguments given to `indent`. The reformatting process introduces little delay, typically one or two seconds for a few hundred line source file. Unfortunately, the reformatted source must be reparsed and cross-referenced by `xc` as the position (line and column) of functions and variables within the file may have changed. When the source window is destroyed, the reformatted source may be saved under the original filename after confirmation.

4. Some Implementation Details

One of the first responsibilities of `xc` is to build a symbol table of all "available" object and archive files presented via command line arguments and environment variables. Armed with this information, `xc` can report the location (site of definition) of functions and global identifiers external to the provided source files. When identifiers are selected with the mouse, and their source code definition cannot be found, the pre-built symbol tables are searched in the same order as does the link editor, `/bin/ld`, and the dynamic linking object, `ld.so`. In support of this, `xc` recognizes two command line arguments of the linker: `-L` to augment the list of standard directories in which to search for libraries, and `-l` to specify abbreviated library names. The default directories `/lib`, `/usr/lib` and `/usr/local/lib` anchor the list of directories. Directory names provided in the `LD_LIBRARY_PATH` environment variable prefix the search.

Global function and variable names are extracted from the namelist of each library and collated into a hash table with 2^{10} buckets. This appears to give a reasonable distribution for the symbol table search. Both existent and non-existent identifiers are found (or not) in 2-7 probes of the hash buckets. As a representative example, symbol table construction for the 981, 2054 and 815 externals in the `libc`, XView and X11 libraries, respectively, is performed in times of `real=2089ms,usr=580ms,sys=1040ms`.

The next task of `xc` is to build a concordance of all available UNIX manual entries. During the execution of `xc`, the selection of an identifier reports if a UNIX manual entry is available for that identifier (typically a library function). Rather than searching for the appropriate manual at selection time, which typically takes 45-50 probes of the filesystem for a successful search and 75-85 for an unsuccessful one (with the consequent NFS requests), *all* "available" manual entry filenames are hashed into a bitmap. This results in the corresponding (identical) number of hash table probes for a successful search, but only 1-3 filesystem lookups (and of course no filesystem lookups if all the 75-85 hash table probes fail). This provides an almost optimal scheme, as identifiers such as `exit` appear in sections `1, 2` and `3` of the manual.

The UNIX environment variable `MANPATH` is first sought from the invoking environment (or `/usr/man` in its absence). The filename of each manual in sections `1, 2, 3, 5, 7` and `l` (together with 9 additional sub-suffixes) is hashed into bitmap of 2^{18} entries using a prime-conguential hashing algorithm. With the simplistic `MANPATH` of only `/usr/man`, the 1965 manuals on our SunOS 4.1.3 systems are collated in the hash table with only a 0.20% collision frequency, in

the times of `real=763ms,usr=50ms,sys=90ms`. With the more significant (and typical) `MANPATH=/usr/man:/usr/local/man:/usr/local/openwin/man:/usr/contrib/man`, the 3217 manuals are collated with a 0.53% collision frequency, in times of `real=3441ms,usr=90ms,sys=210ms`.

One problem encountered in the prototyping of `xc` was the implementation of scrolled canvases in the XView toolkit. Under XView, canvas lengths (in pixels) are recorded as **`short`** (16 bit) integers and, hence, a file displayed in, say, a 14-point font may be a maximum of about 2340 lines before a problem arises (in practice the canvas "wraps around" and the file contents is replicated). While this limit is rarely exceeded, the 4300 line header file `<X11/Xlib.h>` presents an example of the problem. To overcome this, a new form of object supporting very large canvases was constructed in which the unit of scrolling is a text line and not a pixel.

5. Related and Further Work

The ideas addressed and implemented by `xc` are certainly not new and other *commercial* implementations provide similar features and more. Notable examples include Sun's unbundled SPARCcompiler with its source code browser `sbrowser`, the products `CodeCentre` and `SabreC` on many disparate UNIX platforms and `Turbo-C` and `Quick-C` under DOS environments. Each of these implementations attempts to integrate its source navigation with their editors and, in some cases, their debuggers. `xc` is far less ambitious in only attempting code navigation.

`xc` only demonstrates what is possible within a single tool but at only 3200 lines of C, `lex` and `yacc` there are still many ideas to incorporate. Presented here are a few items from a "wish-list" of additions that will hopefully be made to `xc`:

- The responsibilities of both `cpp` and the parser of a full C compiler need to be incorporated into `xc` to provide more accurate cross-referencing of source code. At present, selection of an identifier such as `stderr` is reported as an unknown external identifier rather than the pattern `(&_iob[2])` from `<stdio.h>`. Similarly, "identifiers" such as `MAXPATHLEN` could be correctly described in terms of their numeric values.

- Complete inclusion of both preprocessing and parsing will enable function prototypes to be correctly reported with full typing information – and perhaps checked. This facility is already available in the Free Software Foundation's `gcc` in which external function usage is checked against prototypes read from standard specification files. At present, `xc` only reports the location of an external function from within an object file or archive and the returned type or argument type information cannot be obtained.

- It is hoped more informative explanations of identifiers can be provided. One possibility includes enabling the selection of short blocks of source code text with the mouse (akin to cutting before pasting) and then describing that selection. This addition is feasible if the actions of the utility `cdecl` can be incorporated into `xc` – variable declarations could then be described in terse English.

- Although the execution speed of `xc` on a moderately loaded workstation is appreciable, there is scope for reducing the speed of collecting relatively stable information. Examples include the global library archives such as `libc.a` and the X-window archives which remain unchanged for months. Similarly, Tichy and Litman

window presenting the requested header file.

The other button on each source window, labeled `Reformat`, displays a selection panel from which about twenty attributes may be modified to control the appearance of the source file window. Most attributes are boolean, and are selected using non-exclusive buttons. Numeric attributes are controlled with integral sliders. The reformatting of the source code is, of course, not performed by `xc` itself. Instead, the current contents of the source window is presented as standard input to the `indent` utility; its standard output is redirected to a file and the contents of this file are displayed on the original source window. The attribute buttons and sliders control the command line arguments given to `indent`. The reformatting process introduces little delay, typically one or two seconds for a few hundred line source file. Unfortunately, the reformatted source must be reparsed and cross-referenced by `xc` as the position (line and column) of functions and variables within the file may have changed. When the source window is destroyed, the reformatted source may be saved under the original filename after confirmation.

4. Some Implementation Details

One of the first responsibilities of `xc` is to build a symbol table of all "available" object and archive files presented via command line arguments and environment variables. Armed with this information, `xc` can report the location (site of definition) of functions and global identifiers external to the provided source files. When identifiers are selected with the mouse, and their source code definition cannot be found, the pre-built symbol tables are searched in the same order as does the link editor, `/bin/ld`, and the dynamic linking object, `ld.so`. In support of this, `xc` recognizes two command line arguments of the linker: `-L` to augment the list of standard directories in which to search for libraries, and `-l` to specify abbreviated library names. The default directories `/lib`, `/usr/lib` and `/usr/local/lib` anchor the list of directories. Directory names provided in the `LD_LIBRARY_PATH` environment variable prefix the search.

Global function and variable names are extracted from the namelist of each library and collated into a hash table with 2^{10} buckets. This appears to give a reasonable distribution for the symbol table search. Both existent and non-existent identifiers are found (or not) in 2-7 probes of the hash buckets. As a representative example, symbol table construction for the 981, 2054 and 815 externals in the `libc`, XView and X11 libraries, respectively, is performed in times of `real=2089ms,usr=580ms,sys=1040ms`.

The next task of `xc` is to build a concordance of all available UNIX manual entries. During the execution of `xc`, the selection of an identifier reports if a UNIX manual entry is available for that identifier (typically a library function). Rather than searching for the appropriate manual at selection time, which typically takes 45-50 probes of the filesystem for a successful search and 75-85 for an unsuccessful one (with the consequent NFS requests), *all* "available" manual entry filenames are hashed into a bitmap. This results in the corresponding (identical) number of hash table probes for a successful search, but only 1-3 filesystem lookups (and of course no filesystem lookups if all the 75-85 hash table probes fail). This provides an almost optimal scheme, as identifiers such as `exit` appear in sections `1, 2` and `3` of the manual.

The UNIX environment variable `MANPATH` is first sought from the invoking environment (or `/usr/man` in its absence). The filename of each manual in sections `1, 2, 3, 5, 7` and `1` (together with 9 additional sub-suffixes) is hashed into bitmap of 2^{18} entries using a prime-conguential hashing algorithm. With the simplistic `MANPATH` of only `/usr/man`, the 1965 manuals on our SunOS 4.1.3 systems are collated in the hash table with only a 0.20% collision frequency, in

the times of `real=763ms,usr=50ms,sys=90ms`. With the more significant (and typical) `MANPATH=/usr/man:/usr/local/man:/usr/local/openwin/man:/usr/contrib/man`, the 3217 manuals are collated with a 0.53% collision frequency, in times of `real=3441ms,usr=90ms,sys=210ms`.

One problem encountered in the prototyping of `xc` was the implementation of scrolled canvases in the XView toolkit. Under XView, canvas lengths (in pixels) are recorded as **`short`** (16 bit) integers and, hence, a file displayed in, say, a 14-point font may be a maximum of about 2340 lines before a problem arises (in practice the canvas "wraps around" and the file contents is replicated). While this limit is rarely exceeded, the 4300 line header file `<X11/Xlib.h>` presents an example of the problem. To overcome this, a new form of object supporting very large canvases was constructed in which the unit of scrolling is a text line and not a pixel.

5. Related and Further Work

The ideas addressed and implemented by `xc` are certainly not new and other *commercial* implementations provide similar features and more. Notable examples include Sun's unbundled SPARCcompiler with its source code browser `sbrowser`, the products `CodeCentre` and `SabreC` on many disparate UNIX platforms and `Turbo-C` and `Quick-C` under DOS environments. Each of these implementations attempts to integrate its source navigation with their editors and, in some cases, their debuggers. `xc` is far less ambitious in only attempting code navigation.

`xc` only demonstrates what is possible within a single tool but at only 3200 lines of C, `lex` and `yacc` there are still many ideas to incorporate. Presented here are a few items from a "wish-list" of additions that will hopefully be made to `xc`:

- The responsibilities of both `cpp` and the parser of a full C compiler need to be incorporated into `xc` to provide more accurate cross-referencing of source code. At present, selection of an identifier such as `stderr` is reported as an unknown external identifier rather than the pattern `(&_iob[2])` from `<stdio.h>`. Similarly, "identifiers" such as `MAXPATHLEN` could be correctly described in terms of their numeric values.

- Complete inclusion of both preprocessing and parsing will enable function prototypes to be correctly reported with full typing information – and perhaps checked. This facility is already available in the Free Software Foundation's `gcc` in which external function usage is checked against prototypes read from standard specification files. At present, `xc` only reports the location of an external function from within an object file or archive and the returned type or argument type information cannot be obtained.

- It is hoped more informative explanations of identifiers can be provided. One possibility includes enabling the selection of short blocks of source code text with the mouse (akin to cutting before pasting) and then describing that selection. This addition is feasible if the actions of the utility `cdecl` can be incorporated into `xc` – variable declarations could then be described in terse English.

- Although the execution speed of `xc` on a moderately loaded workstation is appreciable, there is scope for reducing the speed of collecting relatively stable information. Examples include the global library archives such as `libc.a` and the X-window archives which remain unchanged for months. Similarly, Tichy and Litman

[TIC86][LIT93] report reductions of between 25%-65% in compilation and linking times when the standard C `#include` files are pre-compiled.

- No consideration has yet been given to the effective use of colour in `xc` and, again, there is a wealth of material reported by Baecker and Marcus [BM90] and in *SIGCHI* and the human-factors' literature. Examples include highlighting of identifiers in different colours, perhaps indicating their lexical scope. Similarly, selection of a C keyword such as **`break`**, or **`continue`** could result in its enclosing syntactic block being highlighted. Overuse of colour can, however, be as detrimental as it can be effective.

Unfortunately, the list could go on. The Appendix presents an example of `xc` under execution. The leftmost window presents the initial source file `xc.c` when `xc` was invoked with the command line in Section 3. The external function `getenv` has been selected, with the left mouse button and its UNIX manual entry requested with the right mouse button. The manual appears in the frontmost window. The C preprocessor directive `#include "xc.h"` has also been selected from the initial source window, and is presented in the mostly hidden window.

6. References

The interested reader is also referred to the extensive bibliography provided by Baecker and Marcus [BM90] which addresses far more than just the typographic presentation of programs.

[BLO93] A. Bloesch, *Aesthetic Layout of Generalized Trees*, Software – Practice and Experience, Vol 23, No 8., Aug 1993, pp817-827.

[BM90] R.M. Baecker and A. Marcus, *Human factors and typography for more readable programs*, ACM Press, 1990.

[CM89] D. Comer and S. Munson, *Operating System Design Volume I: The XINU Approach*, (Macintosh edition), Prentice-Hall, 1989.

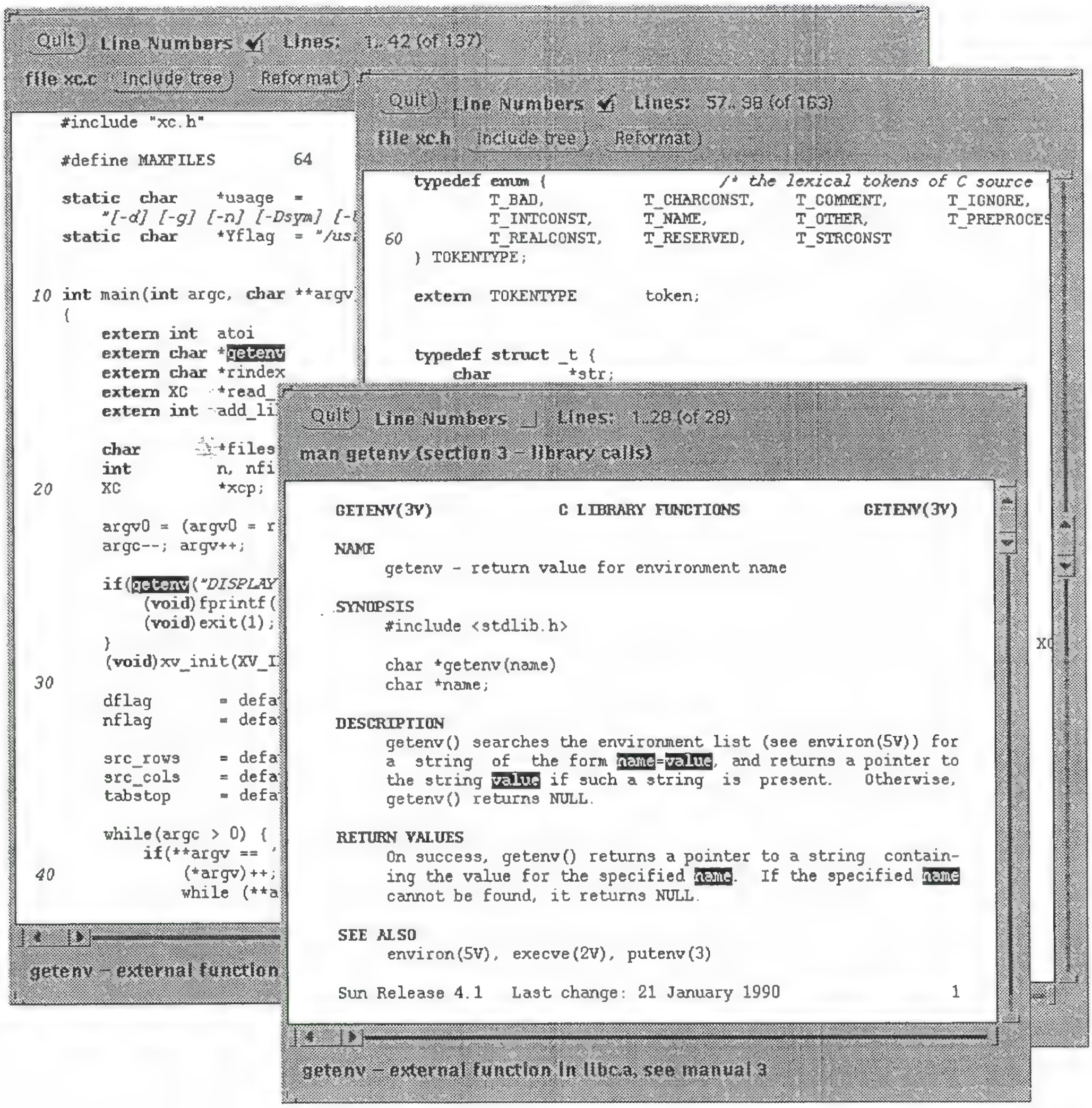
[HTJ91] W.R. Hutchinson, J.H. Tischer, C.A. Johnson, D.A. Dargel and B.A. Rudolph, *A Source Code Navigation Tool for the XINU Operating System*, in Proceedings of the 22nd ACM Computer Science Education Conference '91, San Antonio, Mar 1991, pp304-308.

[LIT93] A. Litman, *An Implementation of Precompiled Headers*, Software – Practice and Experience, Vol 23, No 3., Mar 1993, pp341-350.

[TIC86] W. Tichy, *Smart Recompilation*, ACM Transactions on Programming Languages and Systems, Vol 8, No 3., Mar 1986, pp273-291.

[VAU80] J.G. Vaucher, *Pretty-printing of Trees*, Software – Practice and Experience, Vol 10, No 6., Jun 1980, pp553-561.

Appendix – an example of xc under execution



Appendix – an example of `xc` under execution

Quit } Line Numbers ✓ Lines: 1 42 (of 137)

file xc.c Include tree } Reformat }

```
    #include "xc.h"

    #define MAXFILES        64

    static  char    *usage  =
        "[-d] [-g] [-n] [-Dsym] [-
    static  char    *Yflag  = "/us

10  int main(int argc, char **argv
    {
        extern int  atoi
        extern char *getenv
        extern char *rindex
        extern XC   *read_
        extern int  add_li

        char        *files
        int         n, nfi
20      XC          *xcp;

        argv0 = (argv0 = r
        argc--; argv++;

        if(getenv("DISPLAY
            (void)fprintf(
            (void)exit(1);
        }
        (void)xv_init(XV_I
30
        dflag       = defa
        nflag       = defa

        src_rows    = defa
        src_cols    = defa
        tabstop     = defa

        while(argc > 0) {
            if(**argv == '
40              (*argv)++;
                while (**a
```

Quit } Line Numbers ✓ Lines: 57 98 (of 163)

file xc.h Include tree } Reformat }

```
    typedef enum {                  /* the lexical tokens of C source
            T_BAD,          T_CHARCONST,    T_COMMENT,      T_IGNORE,
            T_INTCONST,     T_NAME,         T_OTHER,        T_PREPROCES
60          T_REALCONST,    T_RESERVED,     T_STRCONST
    } TOKENTYPE;

    extern  TOKENTYPE       token;


        typedef struct _t {
            char        *str;
```

Quit } Line Numbers ✓ Lines: 1 28 (of 28)

man getenv (section 3 – library calls)

```
GETENV(3V)              C LIBRARY FUNCTIONS              GETENV(3V)

NAME
     getenv - return value for environment name

SYNOPSIS
     #include <stdlib.h>

     char *getenv(name)
     char *name;

DESCRIPTION
     getenv() searches the environment list (see environ(5V)) for
     a  string  of  the form name=value, and returns a pointer to
     the string value if such a string  is  present.   Otherwise,
     getenv() returns NULL.

RETURN VALUES
     On success, getenv() returns a pointer to a string  contain-
     ing the value for the specified name.  If the specified name
     cannot be found, it returns NULL.

SEE ALSO
     environ(5V), execve(2V), putenv(3)

Sun Release 4.1   Last change: 21 January 1990                1
```

getenv – external function

getenv – external function in libc.a, see manual 3

Unix on Steroids: Doping Control at Barcelona '92

Robert Ewin
Computer Services Centre
Australian National University

AUUG Canberra Chapter
Fifth Annual Canberra Conference and Workshops
February 1994

Introduction

Doping control of athletes in Olympic competition first began at the summer and winter games of 1968, celebrated in Mexico City and Grenoble respectively. Dope testing has occurred at all subsequent summer and winter games, which are celebrated at four yearly intervals. From 1994, summer and winter games are no longer being held in the same year.

Doping control laboratories are accredited by the Medical Subcommission of the International Olympic Committee. Currently there are 22 IOC accredited laboratories, including one in Sydney, two in the United States (Los Angeles and Indianapolis), two in Spain (Madrid and Barcelona) and one in Oslo, Norway. As yet, there is no accredited laboratory in Atlanta.

Doping control for the 1992 Barcelona Olympic Games was performed by the Pharmacology Department of the Municipal Medical Research Institute (IMIM) in Barcelona. 1887 urine samples were processed in 15 days of competition, generating about 10 gigabytes of data. Many new computing technologies were employed in Olympic competition for the first time. Among these were the use of Unix, a Laboratory Information Management System (LIMS), network connection of analytical instruments, the Oracle data base system, and some sophisticated reporting modules developed within the laboratory.

The person responsible for the computing component of the laboratory during 1991 and 1992 was an Australian.

The Doping Control Laboratory

About the same time that the 1992 Olympic Games were awarded to Barcelona in 1986, IMIM sought and gained doping control accreditation with the IOC. Not only does the process of accreditation require that each laboratory successfully complete a series of tests supplied by the IOC, but accredited laboratories are also subject to annual certification. Failure to complete annual tests results in the loss of accreditation, as occurred with the laboratory in Brisbane which was responsible for doping control at the 1982 Commonwealth Games in that city. Nomination of a laboratory for IOC accreditation usually also involves political considerations.

IMIM is a medical research institute with six departments and a staff of about 120, about half of whom are postgraduate research fellows. It was the Department of Pharmacology and Toxicology within IMIM which effectively became the doping control laboratory for Barcelona '92. Normally, it is a small department of between 20 and 30 staff, and remained so until the beginning of 1991, about 18 months before the Games commenced in July 1992.

At the end of 1990, contracts were signed with Hewlett-Packard for the supply of a significant number of analytical instruments and associated work stations. At the same time, a LIMS system, including substantial software development and an Oracle run time licence, was purchased from a Spanish software house. The appointment of a computer programmer to oversee the installation, configuration and development of the software, and of the network, occurred in January 1991.

The department grew to about 45 by the end of 1991, and for two months before and during the Games, an additional 35 vacation students were added to the team. There were 80 staff working a number of overlapping shifts, 24 hours per day, during the 15 days of competition.

Analytical Process

Until the winter Olympics of 1994, doping control had only involved the testing of urine samples from humans and horses. In Norway in 1994, a small number of blood samples will be tested, primarily for the determination of sex.

In urine dope control, two samples of 50 ml each, called the A and B samples, are supplied at the end of designated events by the three medal winners, and usually a small random sample of other athletes. The A sample is used for analysis, and the B sample is used for confirmation in the event of a positive result. A sample is positive if it shows traces of one or more of a set of IOC banned substances, or in the case of some substances (eg. caffeine), a concentration higher than the IOC defined maximum limit. In such cases, the Medical Subcommission is notified, and they in turn notify the team manager and the athlete concerned. Analysis is then repeated with the B sample in the presence of the athlete, team representative and members of the IOC. This analysis is performed manually, and with more sensitive equipment than the original sem automatic analysis of the A sample. A positive result is usually followed by sanctions against the athlete, or team, concerned.

The IOC Medical Subcommission also secretly includes a number of positive samples amongst those supplied by athletes. The purpose of these positive samples is to test the correct functioning of the laboratory during competition. Five spiked samples were included by the IOC, and detected by the Barcelona laboratory, four of these in the first two days of competition.

Banned Substances

Banned substances are divided into a number of categories depending on their pharmacological properties, and their affect on competitors. The five principal categories are Stimulants (eg. Amphetamines), Narcotics and Beta Blockers (eg. Morphine, Codeine), Caffeine, Anabolic Steroids (eg. Stanozolol, Testosterone) and Diuretics (eg. Mesocarb). Beta blockers have a tranquillising effect and would usually be taken by shooters or archers; Stimulants may be taken to sharpen reaction, such as in fencing; Steroids are taken by lifters, sprinters and anyone requiring additional muscle bulk; Diuretics aid rapid weight loss (boxers) and act as a masking agent for other drugs.

The IOC lists several hundred drugs, classified under the above headings, as being banned substances. The doping control laboratory is expected to be able to detect the presence of any one of these drugs, and almost always tests all athletes for all drugs, although this usually makes little sense - steroids would rarely be taken by long distance runners, and beta blockers would not normally be taken by sprinters.

Analytical Equipment

After chemical pre-processing, each sample is divided into a number of sub-samples of between one and 5 ml each. These are then analysed in batches using equipment appropriate to the detection of each of the major drug classes. Each batch of sub-samples usually contains a maximum of twenty vials, to which is added a number of blank, standard or positive samples for calibration purposes.

Batches of samples are then processed, according to the class of drugs being screened, using Gas Chromatography (GC), Gas Chromatography with Mass Spectrometry (GC/MS), or High Pressure Liquid Chromatography (HPLC). All analytical instrumentation was supplied by Hewlett-Packard, world leaders in Gas and Liquid Chromatography, and Mass Spectrometry

During the Barcelona Games, six Hewlett-Packard (HP) GC instruments controlled by Vectra (MS-DOS) systems were used to screen for Stimulants; six HP GC/MS instruments with 9000/345 (Unix) systems screened for Narcotics and Beta Blockers; three HP HPLC instruments with 9000/300 (Pascal) systems screened for Caffeine; six GC/MS instruments (Unix) screened for Anabolic Steroids; and three HPLC (Pascal) instruments screened for Diuretics. Two larger GC/MS/LC instruments, both controlled by Unix work stations, were used for confirmation tests, and a smaller GC/MS (Unix) system was used for doping control of horses.

Depending on the screening process and instrumentation, and the level of sensitivity required, each sample can take between a few minutes and one hour to process. They each produce data files ranging in size from several kilobytes to several megabytes, and hard copy reports of from one to twenty pages. The number of instruments involved in doping control during Barcelona '92 was calculated on the number of samples and the assumption of almost round the clock operation.

Computing Equipment

Each analytical instrument is controlled by a personal computer or work station. In the case of Hewlett-Packard analytical equipment, Gas Chromatography systems are driven by a HP Vectra (PC) using DOS and Windows; GC/MS instruments are controlled by HP 9000/345 Unix work stations with DAT and 600 megabyte disk; and HPLC instruments use an older HP 9000/300 "Pascal" work station. This latter machine runs a stand alone application, probably originally written in Pascal, hence the name. There is no operating system as such for these machines.

In addition to the six Vectras, 15 Unix work stations and six Pascal systems described above, the laboratory purchased a HP 9000/832 system with 900 megabytes of disk and DAT drive to support a Laboratory Information Management System (LIMS), based on an Oracle data base. Hewllet-Packard also lent a 9000/400 Unix file server with 1200 megabytes of disk and DAT, for the duration of the Games. Each of these PC's and work stations was configured with its own HP LaserJet II printer, and a hand held bar-code reader for sample identification. Some instruments were also configured with carousal type bar code readers, and robotic arms for sample processing.

Apart from computer systems associated with instruments, the LIMS and file server, five 19 inch colour X terminals, and five ascii terminals were used for remote access to the instruments, and about 10 IBM compatible PC's were used for document processing.

All equipment was connected to an ethernet based Local Area Network, which was in turn connected via a Bridge to the LAN of the Institute. IMIM is not connected to the Internet, but does have an X.25 link which is used to gateway electronic mail. Their address is ***imim.es***.

Software

Vectra PC's used MS-DOS, Windows, Hewlett-Packard's *ChemSystem* application for control and analysis of GC instruments and data, and a third party implementation of TCP/IP for automatic file transfer (ftp) of data to the server and LIMS systems.

Unix work stations used HP-UX, Motif, HP's *ChemSystem* for control and analysis of GC/MS instruments and data, and the standard HP-UX TCP/IP utilities for communication with the LIMS and to transfer data to the file server.

Pascal work stations, the oldest of HP's analytical controllers, runs a stand alone version of the *ChemSystem* application, and a third party implementation of TCP/IP to transfer data to the server.

A Laboratory Information Management System called *LABiX* was purchased from a Spanish software house, and tailored for use in doping control. The LIMS is based on an Oracle data base, although only a run-time licence was included with the product. No development was possible with Oracle, and all subsequent in house analysis was done on PC's using dBASE IV.

A data collection, extraction and archiving module was written by the suppliers of *LABiX*, and improved reporting modules were developed within the laboratory. It was the use of *LABiX*, the networking of instruments, and the reporting procedures which generated the greatest interest from the IOC.

Laboratory Information Management System

A LIMS called *LABiX* was used on the main Unix system to control the flow of urine samples through the laboratory. *LABiX* is really an interface to an Oracle data base, and uses a series of textual menus to provide key laboratory personnel, and the screening analysts, with access to the data base.

In addition to the data base component of *LABiX*, several modules were developed especially for doping control. These included the production of bar code labels for batches, vials, tubes and envopaks, and a daemon which extracted key fields from sample data as it arrived (via ftp) from the instruments. Although *LABiX* is called a Laboratory Information Management System, like all similar systems available at the time, it is not capable of unifying its data sources through the network. The so called *Unified Laboratory* that is much discussed in doping control, is not yet a reality.

Target Analysis

In screening procedures using Gas Chromatography, samples are heated and ions are counted as they are emitted. The plot of ions against retention time is known as a chromatogram. Substances are identified by the occurrence of a peak at a particular (known) retention time in the chromatogram.

Gas Chromatography/Mass Spectrometry differs from simple GC by the addition of a count of the mass of each ion emitted. In GC/MS, the presence of a substance depends on the existence of a peak for each of a set of ions at the expected retention time.

High Pressure Liquid Chromatography detection relies on the occurrence of peaks with characteristic ultra violet spectra. Substance detection relies on the visual matching of the UV spectra with those of a library of UV spectra for known substances.

In each situation, the use of known retention times and peak characteristics is known as *target analysis*, and one of the major innovations of the Barcelona laboratory was the refinement it made to the use of target analysis in doping control. The set of highly graphic and detailed screening reports which were designed and developed in the laboratory proved a great aid in detection, and led to substantial improvements in throughput and productivity.

Conclusion

The laboratory worked for 24 hours each day during 15 days of competition to process 1887 urine samples. All five IOC planted positive samples were detected, as were five positive samples belonging to athletes. None of the positive samples detected in competition involved medal winners, and the Barcelona Games were considered relatively clean, due both to the lack of conytroversy, and the relatively small number of positive samples encountered.

In terms of data, almost 40 DAT tapes were written, amounting to about 15 gigabytes of sample and report data, duplicated a number of times. It is also estimated that 100,000 pages of reports were printed.

Procedimiento 4B: Esteroides - Fracción Combinada

Farmacologia y Toxicologia, I.M.I.M.

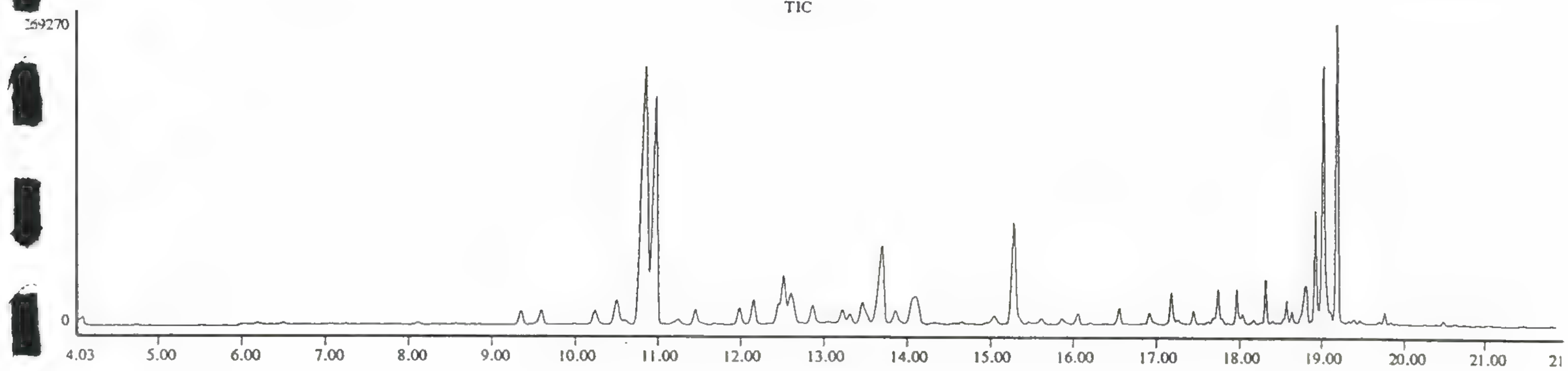
Fichero: 4B2020201002.d Ordenador: hpux9

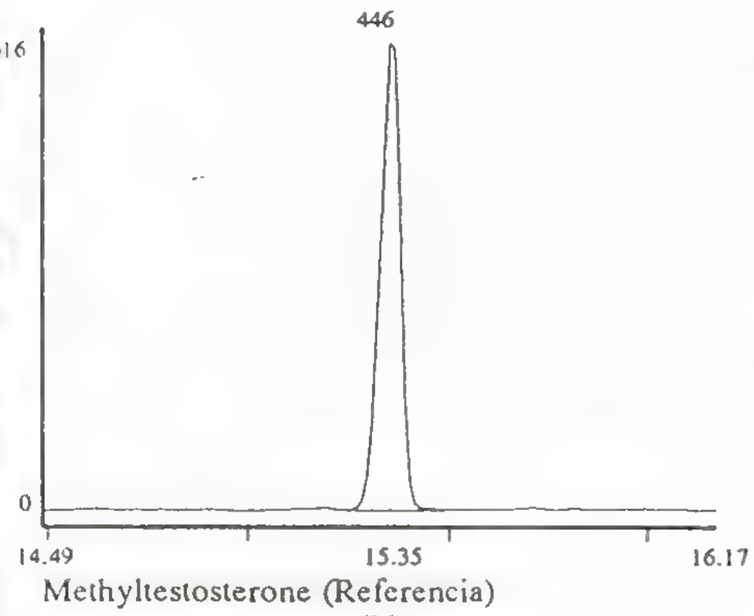
Muestra: ORP1L924B202 Lote: L924B202

Codigo Leido: L924B202

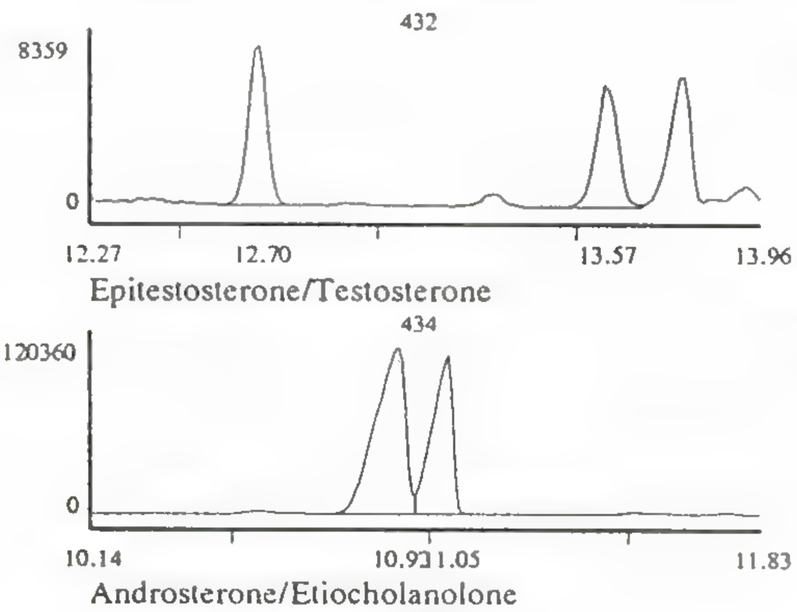
Instrumento: HP5971 Fecha: Sat Sep 05 92 02:43:54 PM

Indice de Sequencia: 1 Posicion Bandeja: 2 Numero de Replicado: 1

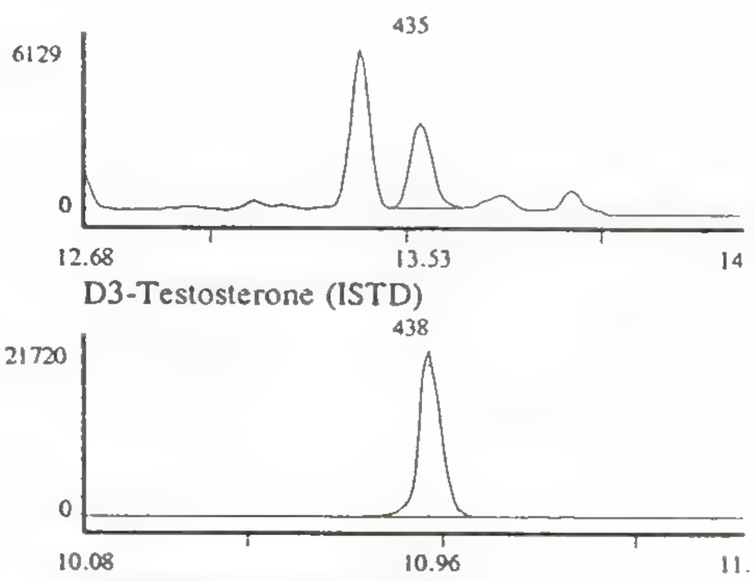




Methyltestosterone (Referencia)



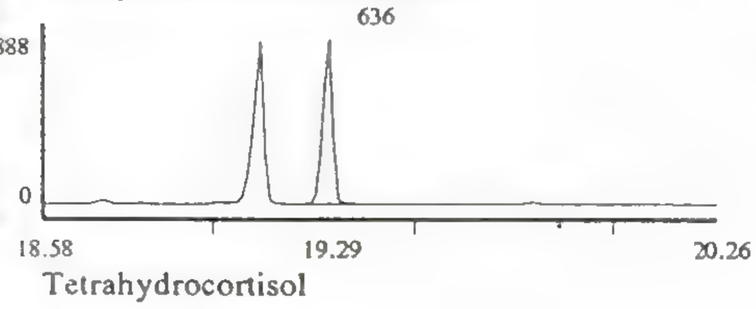
Epitestosterone/Testosterone



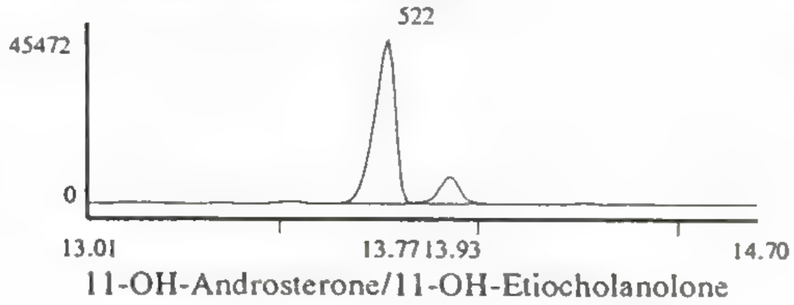
D3-Testosterone (ISTD)

Androsterone/Etiocholanolone

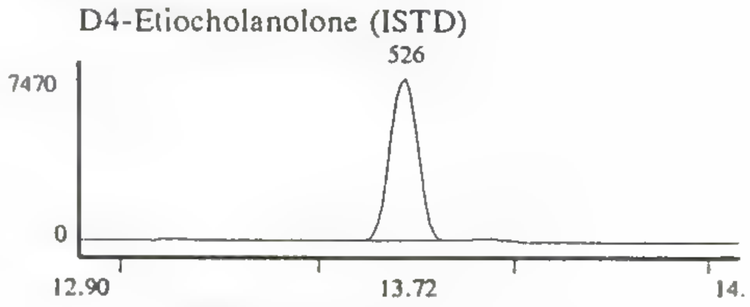
D4-Etiocholanolone (ISTD)



Tetrahydrocortisol



11-OH-Androsterone/11-OH-Etiocholanolone



D4-11-OH-Androsterone (ISTD)

SUSTANCIA	T.R.(min) Esp.	T.R.(min) Real	FACTOR RESPUESTA	AREA	ALTURA	CONC (ng/ml)
Methyltestosterone	15.33	15.35		169308	47467	500.00
Testosterone	13.56	13.57	22.11	24069	6205	43.18
Epistestosterone	12.68	12.70	26.51	26693	8140	57.42
D3-Testosterone	13.53	13.53		12323	3145	20.00
Androsterone	10.84	10.92	577.80	631939	120196	4225.31
Etiocholanolone	11.13	11.05	442.03	408912	113813	2091.64
D4-Etiocholanolone	10.93	10.96		86416	21626	500.00
11-OH-Androsterone	13.77	13.77	225.60	164423	45353	1029.21
11-OH-Etiocholanolone	13.94	13.93	283.03	27513	7420	216.06
D4-11-OH-Androsterone	13.74	13.72		36041	7388	240.00
Tetrahidrocortisol	19.42	19.29	2383.08	253416	149698	3566.94
Testo./Epitesto.				0.90	0.76	0.75
Andro./Etio.				1.55	1.06	2.02
Testo./Andro.				0.04	0.05	0.01
Testo./Etio.				0.06	0.05	0.02

There is no doubt that the use of Unix and MS-DOS to control analytical instruments, the implementation of a LIMS, and the use of a LAN to connect systems, proved to be one of the great successes of the doping control laboratory. Furthermore, although both traditional report generation, and the system of improved target analysis, were jointly used, future doping control laboratories will require sophisticated target analysis procedures similar to those developed in Barcelona.

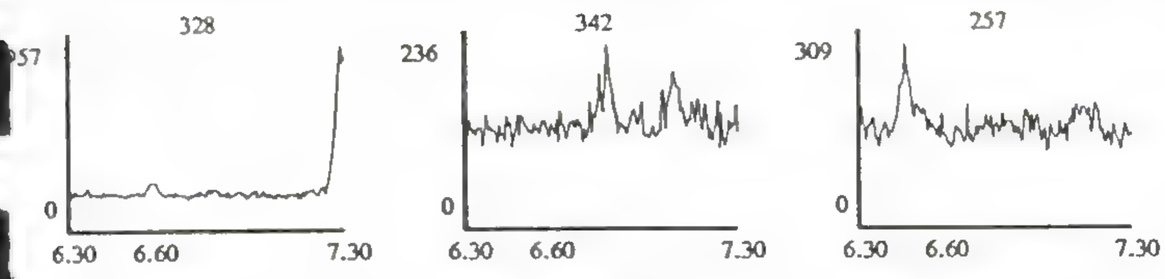
One week after the Games, the size of the laboratory had reduced to about 40 staff, and in September 1992 it processed 200 samples for the Para-Olympic Games. Six positive samples were detected, including one in the gold medal winning United States basketball team. IMIM currently does little doping control, the vast majority of such testing in Spain being performed by a government subsidised laboratory in Madrid. The institute is still IOC accredited, and the Head of the Department of Pharmacology is a current member of the IOC Medical Subcommission.

Procedimiento 4B: Esteroides - Fracción Combinada

Farmacologia y Toxicologia, I.M.I.M.

Fichero: 4B2020201002.d Ordenador: hpux9

Muestra: ORP1L924B202 Lote: L924B202



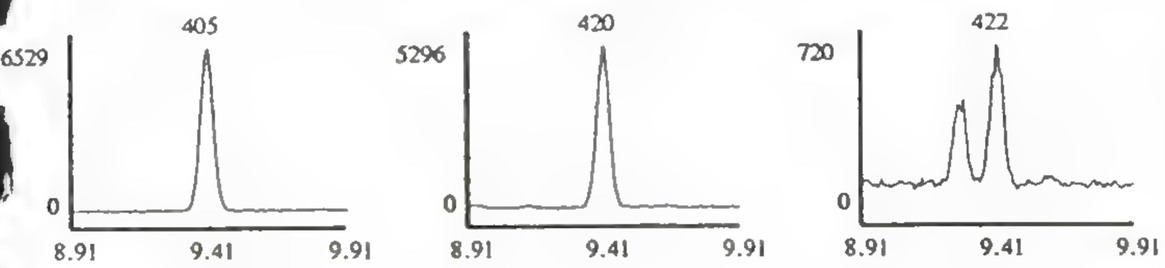
PROBENECID PC

4-(Dipropylsulphamoyl)benzoic acid mono-O-TMS

-

OBSERVACIONES: Observar perfil de endógenos.
Ver también Proc 5.

-



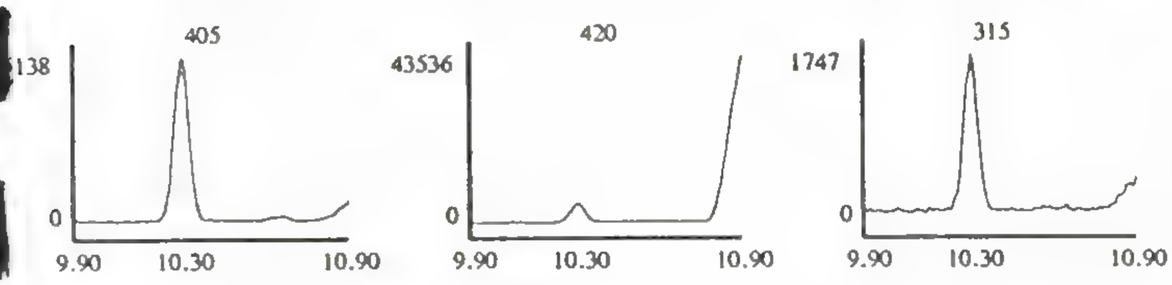
NANDROLONE-Met.1 (19-Norandrosterone)

3α-hydroxy-5α-estran-17-one bis-O-TMS

-

OBSERVACIONES: Es también Norethisterone-Met.2.
m/z 422 abundante de vit.E met.

-



NANDROLONE-Met.2 (19-Noretiocholanolone)

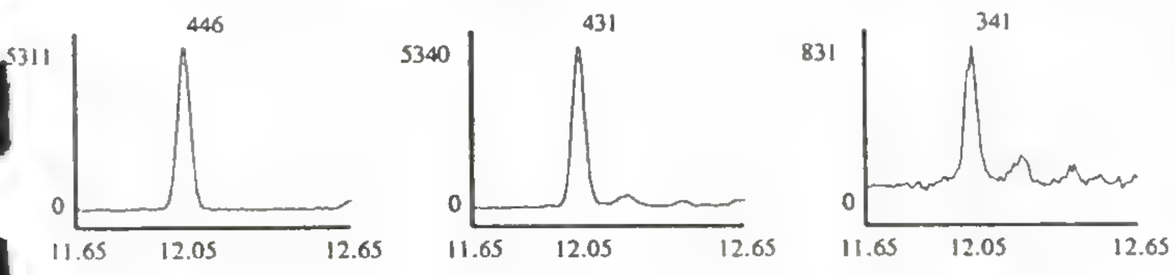
3α-hydroxy-5β-estran-17-one bis-O-TMS

-

OBSERVACIONES: Menor abundancia que Nandrolone Met.1.

-

-

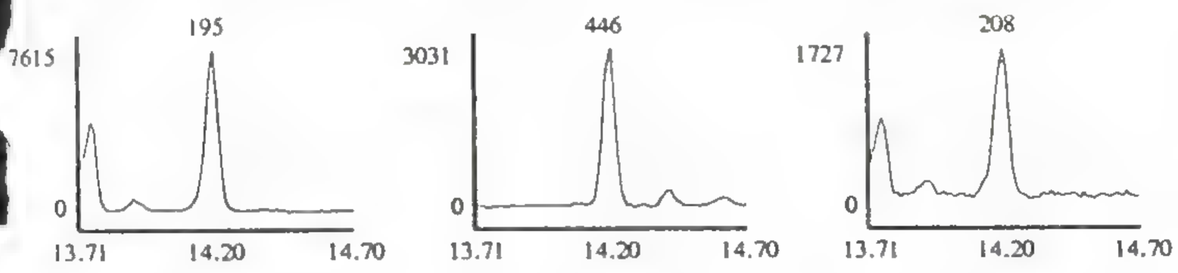


METENOLONE-Met.1

3α-hydroxy-1-methylen-5α-androstan-17-one bis-O-TMS

-

OBSERVACIONES: Coeluye con 4H-Norethisterone (Met.1).
Mismos iones con diferente intensidad.
Ver Nandrolone Met.1.



METENOLONE PC

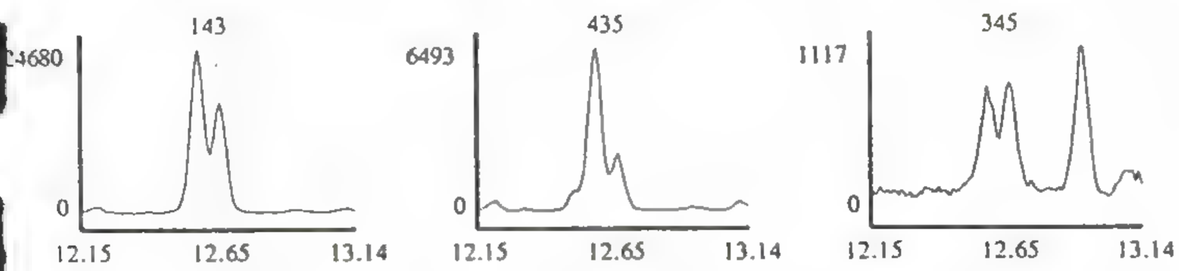
17β-hydroxy-1β-methyl-5β-androst-1-en-3-one bis-O-TMS

-

OBSERVACIONES: -

-

-

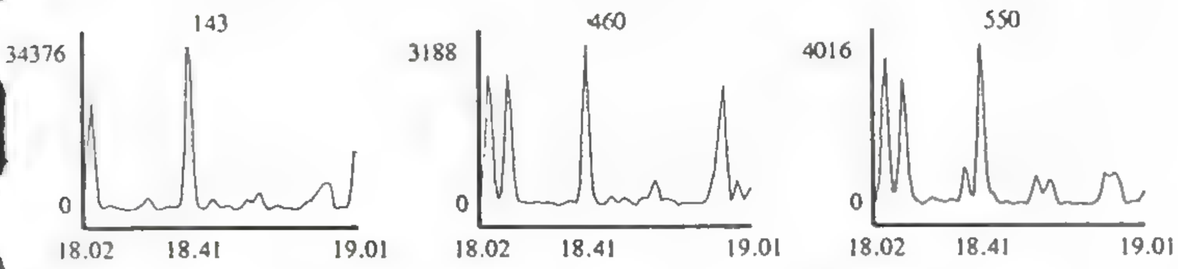


METHYLTESTOSTERONE-Met.1 y 2

17α-methyl-5(α,β)-androstan-3α,17β-diol bis-O-TMS

-

OBSERVACIONES: Met.1 (3α,5α) coeluye con Mesterolone-Met.1.
Es también Oxymetholone-Met.2 y Mestanolone-Met.1.
Met.2 (3α,5β) es Metandienone-Met.3.

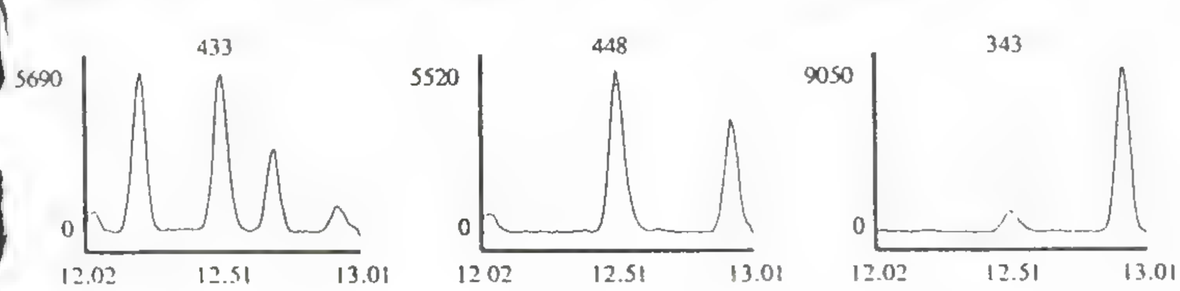


OXYMETHOLONE-Met.1

2-hydroxymethyl-17α-methyl-5α-androstan-3α-4 (?),
17β-triol tris-O-TMS

OBSERVACIONES: Buscar Oxymetholone-Met.2 en ventana de
Methyltestosterone-Met.1.

-



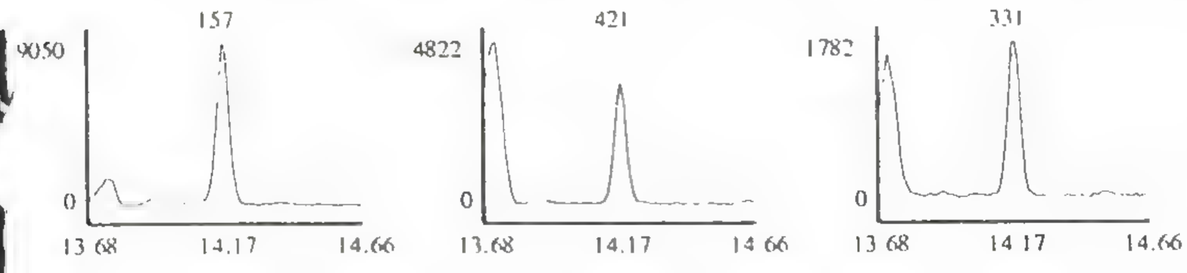
MESTEROLONE-Met.1

3α-hydroxy-1α-methyl-5α-androstan-17-one bis-O-TMS

-

OBSERVACIONES: Ver también ventana de Methyltestosterone-Met.1.
Epitestosterone muy próxima (m/z 433).

-

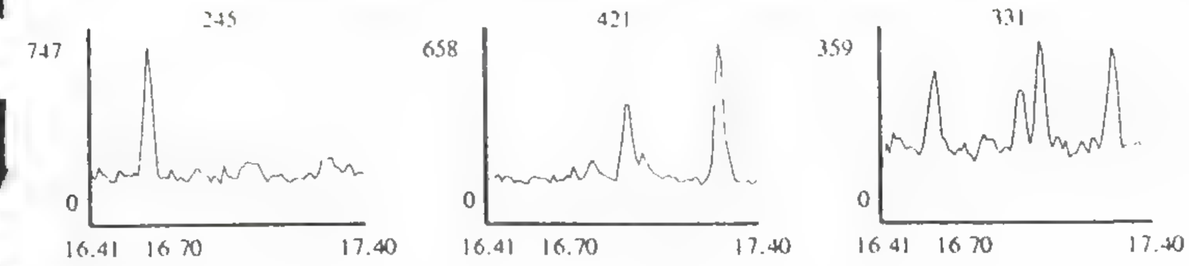


NORETHANDROLONE-Met.1

17α-ethyl-5β-pregnan-3α,17β-diol bis-O-TMS

OBSERVACIONES: -

-



NORETHANDROLONE-Met.2

17α-ethyl-5β-pregnan-3α,17β,20-triol tris-O-TMS

-

OBSERVACIONES: Aparece también otro metabolito no identificado.

-

Practical Applications of Voice / Groupware IP Multicasting

Merik Karman
Defence Service Homes
Department of Veterans Affairs
PO Box 21
Woden ACT 2606
Australia
email: merik@blackadder.dsh.oz.au

Abstract

This paper is intended to introduce UNIX systems administration staff to the practical uses of IP multicasting tools. I will present a brief introduction to IP multicast concepts and demonstrate, using freely available software tools, practical uses of this technology.

This paper is not intended to explain at great technical depth IP multicast, it is intended to make the reader aware of tools that are available today. I will explain how these tools, running over low band-width wide area network (WAN) links can be used to support remote administration staff.

While the Multicast BackbONE (MBONE) is the main high profile use of this technology, I hope to introduce possibilities for intra-organisational communications for campus area networks or private WANs. This discussion is relevant to any organisation utilising UNIX based workstations on a TCP/IP based network and connection to the global Internet is not necessary to use this software.

Introduction

I remember the first time I saw an MBONE session on the Internet, I watched slack jawed as images and sound from a NASA space shuttle mission were multicast from the United States to a Sun Sparcstation on the Australian National University campus.

Tools exist today to make possible the transmission of voice, video and shared white board over existing TCP/IP networks. Multicast uses one Audio and white board can be used on links as slow as 9600 bps dial up with quality being around that of a single side band radio transmission. In this paper I use the term groupware in the title. By way of explanation I mean any tools that allow a group of people to work together despite the distances that they may be apart.

IP multicast is like the proverbial talking dog: its not so much what the dog is saying, its more that the dog can talk at all that is amazing.

Multicast Introduction

MBONE stands for Multicast BackbONE, and originated from the early efforts to multicast Internet Engineering Task Force (IETF) meetings. Multicast is used to deliver audio, video and other groupware applications to many people in the most bandwidth effective manor.(see refs [3],[9]) IP multicast addressing is an Internet standard (Request for Comment RFC-1112[8]) developed by Steve Deering of Xerox Palo Alto Research Centre. Multicast is used because it provides one to many and many to many network delivery services for applications such as video conferencing and audio that need to communicate with several other hosts simultaneously.

The multicast network is a virtual network in that it shares the same physical media as the rest of your IP network. The multicast network is implemented using "tunnels". Tunnelling is a scheme to forward multicast packets between subnets interconnected by IP routers that (typically) do not support IP multicast. This is done by encapsulating the multicast packets inside regular IP packets. I hope the near future will bring multicast routing support in commercial routers which will eliminate the inefficiencies and management headaches of duplicate routers and tunnels.

Network bandwidth is the overriding factor that will influence the use of multicast. The reason why a multicast stream is bandwidth efficient is that one packet can touch all workstations on a network. So a 128 Kbps (kilobits per second) video stream (typically 1-4 frames/sec) uses the same bandwidth whether it is received by one workstation or twenty. Data compression is utilised to reduce bandwidth consumption using compression schemes like PCM, GSM and LPC4 for audio and Joint Photographic Experts Group (JPEG), wavelet based encoding and ISO standard H.261 for video. The multicast stream is controlled to the local subnet unless a "tunnel" exists to ship it downstream. This is necessary because the WAN would soon become saturated without control over the distribution of multicast traffic.

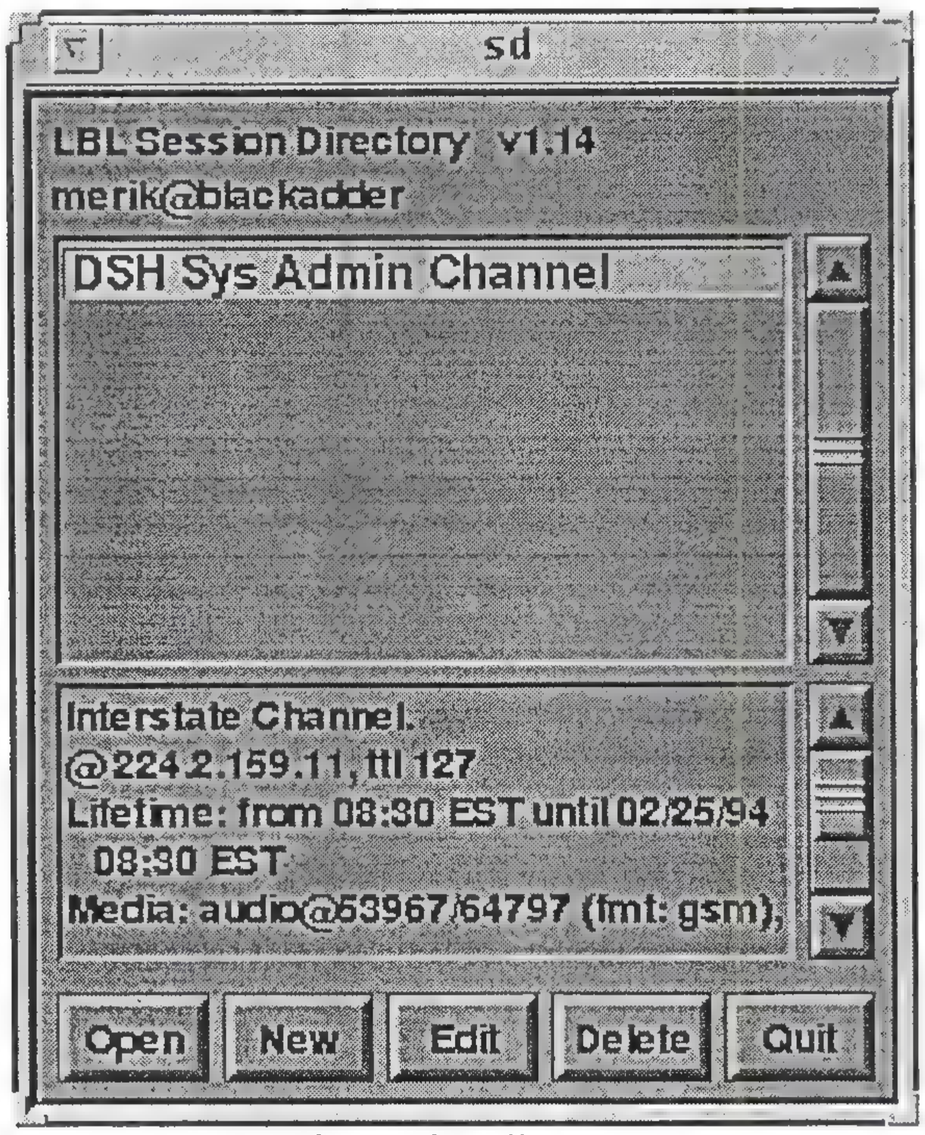
Two methods are employed to control multicast packet distribution. The first is to limit the lifetime of multicast packets and the second is to use sophisticated pruning algorithms to adaptively restrict multicast transmission.

The Tools

The tools used to establish an IP multicast network system are all available via anonymous FTP on the Internet. These tools are typically still in a rapid development phase and should be treated with care in a production environment. They are however of very high standard and can be used by experienced network and systems administrators to do real work now.

sd - session directory

Multicast "events" are announced dynamically by *sd* which displays active multicast groups. *sd* is also used for launching new multicast sessions and automatically selecting unused multicast addresses. *sd* was developed by Steve McCanne and Van Jacobson of Lawrence Berkeley Laboratory, University of California Berkeley.



sd - session directory

vat - visual audio tool

vat is an X-Windows based audio teleconferencing tool written by Van Jacobson (van@ee.lbl.gov) and Steven McCanne (mc-canne@ee.lbl.gov), both of Lawrence Berkeley Laboratory, University of California, Berkeley, CA. *vat* allows users to conduct host-to-host or multi-host audio teleconferences over an internet (multi-host conferences require that the kernel support IP multicast). On most architectures, no hardware other than a microphone is required -- sound I/O is via the built-in audio hardware. It has features such as automatic gain control, push (mouse button) to talk and many different audio format including:

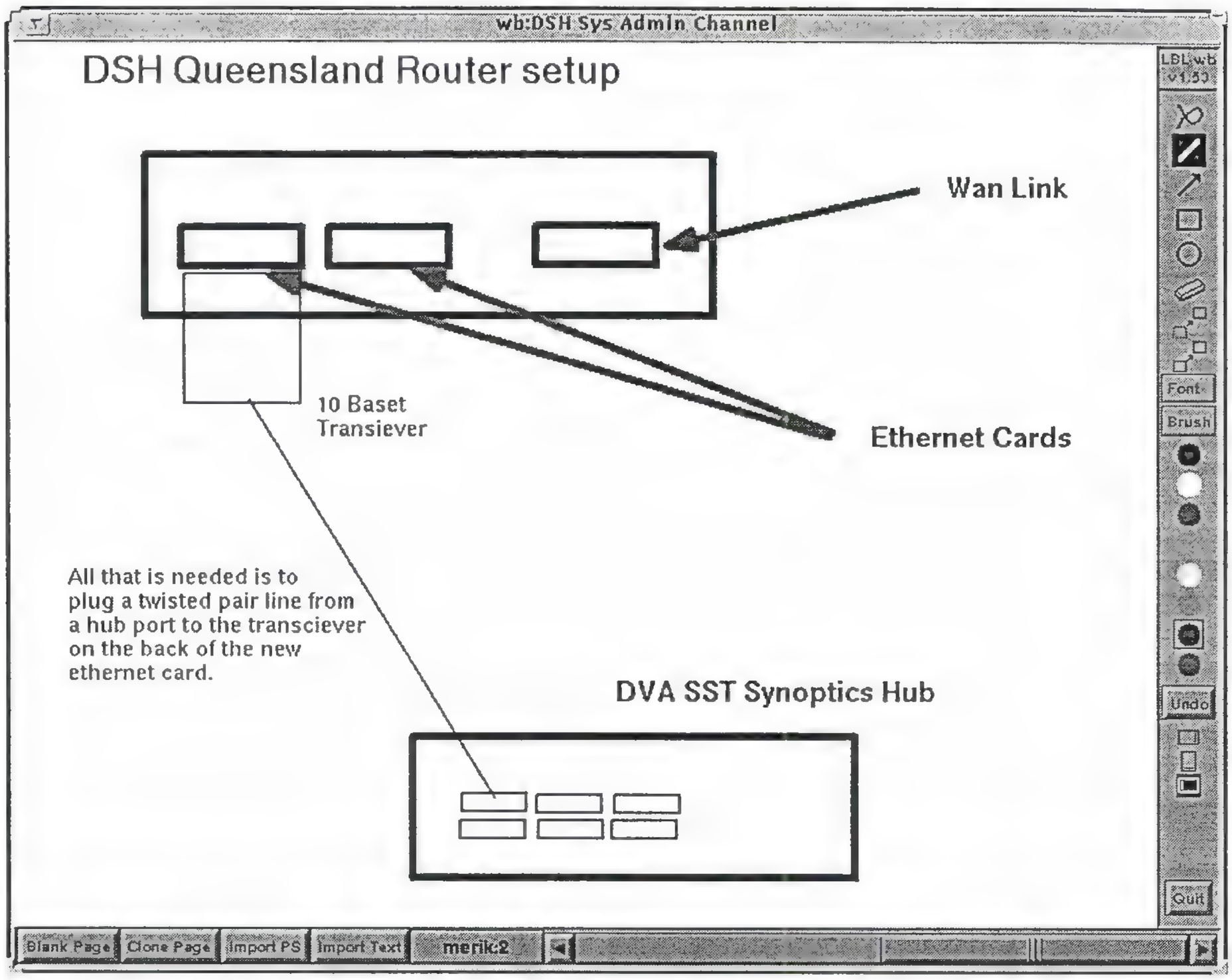
pcm	78Kb/s 8-bit mu-law encoded 8KHz PCM (20ms frames)
pcm2	71Kb/s 8-bit mu-law encoded 8KHz PCM (40ms frames)
pcm4	68Kb/s 8-bit mu-law encoded 8KHz PCM (80ms frames)
idvi	46Kb/s Intel DVI ADPCM (20ms frames)
dvi2	39Kb/s Intel DVI ADPCM (40ms frames)
dvi4	36Kb/s Intel DVI ADPCM (80ms frames)
gsm	17Kb/s GSM (80ms frames)
lpc4	9Kb/s Linear Predictive Coder (80ms frames)

In addition to the above audio formats work is going on with the CELP and LPC 10 audio formats which when complete will consume a tiny 4.8Kb/s and 2.4Kb/s respectively.

vat - visual audio tool

wb - white board

The *wb* application provides a multi-page shared "white board" drawing surface. wb supports basic drawing functions, copy and paste and importing of screen shots, postscript and ASCII text. This application was also written by Van Jacobson (van@ee.lbl.gov) and Steven McCanne (mccanne@ee.lbl.gov), both of Lawrence Berkeley Laboratory, University of California, Berkeley, CA.



wb - white board

nv - net video

nv is the principle tool for video transmission and reception via IP multicast. You will need a video capture board in your workstation to transmit video, but no special hardware is needed to receive video. One to four frames/second video seems slow but in practice it is surprisingly effective when combined with phone quality voice.

Other tools

Additional tools are also available or under development. Winston Dang of the University of Hawaii has created *imm* (Image Multicaster Client), a low bandwidth image server. It is typically used to provide live images of planet Earth from various geostationary satellites at half hour intervals in either visible or infrared spectra. Eve Schooler of University of Southern California (USC)/Information Sciences Institute (ISI) is part of a team developing *mmcc*, a session orchestration tool and multimedia conference control program.

IP Multicast - A Practical Example at Defence Service Homes

The Defence Service Homes (DSH) WAN consists of a central node ACC router with star connections to each major state capital. These links are 64Kbps ISDN semi-permanent links with local ethernets inter-connecting Sun Sparc sytems and PC workstations. Central Canberra Office staff have Sun IPX workstations with remote site administrators using Sun ELC workstations.

I first of all experimented with IP multicast on the local Canberra ethernet using sd, vat and wb. I decided to restrict my experiment to audio and white board as video required a frame capture card, a video camera and significantly more bandwidth. The overriding factor throughout this work was to ensure that user response to the production database applications were not affected. To ensure production systems were not affected I made all traffic between the mroute tunnel machines lowest priority in the routers. It has been working well and when the router has more important traffic to pass it just throws the multicast traffic away resulting in a break up of the audio transmission.

The implementation of IP multicast on the DSH WAN was made easier by utilising the inbuilt support for multicast in Sun Solaris 2.

After trying all the audio formats available under vat I decided on using GSM for long term usage.

Transmitting audio to someone on the workstation five metres from your own is interesting but a little boring so I decided to trial a multicast "tunnel" to the DSH Brisbane office. After a few teething problems the voice of John Bratchford, the systems administrator from Brisbane came booming down at me. We were impressed ! Since that day we have used the system to solve many problems using both voice and white board. The system is taking off and microphones are being ordered to allow all other DSH offices to come on-line.

Systems support staff in Defence Service Homes are using this technology now, to solve our interstate technical communications problems. The screen shot of whiteboard shown earlier in this paper shows how I explained to a remote administrator which port on a router he had to plug a cable. This may seen trivial but the ability to explain with voice and graphical aids solved the problem in very little time. It is like bringing the remote person into your office temporarily and discussing a problem with a pen and paper.

Just how low is low bandwidth

It is said that the wb (white board) application will run over a wet piece of string but what of the other tools? The vat (visual audio tool) can use different compression schemes to consume anywhere from 2400 bps to 78Kbps. The video tools are a different matter and consume typically from 64Kbps to 128Kbps.

What do I need

To start the multicast experiment at your site you need several things:

- **Software** - All of the software you need to start are located on various FTP sites on the Internet. While there are official sites in the United States and Europe I suggest you retrieve the files from the Australian mirror sites for these tools.

  archie.au:/conferencing
  ftp.adelaide.edu.au:/pub/av

- **Workstation** - You currently require a UNIX workstation from Sun, SGI, DEC or HP with ports in progress for Macintosh. No DOS, OS/2, Amiga or Windows versions are currently available although ported tools can be found for 386 machines running the (free) BSD UNIX. The workstation should be equiped with a microphone to use the audio tools.

- **Time** - It will take a good systems administrator one to three days of part time work to establish the basics of an IP multicast system. This time will be taken up by patching and rebuilding the kernel of workstations that do not support multicast standard. This is quite a complex procedure so

you should seek help if you are not versed in such things. Once the kernel is built the majority of the tools are in binary form so take little time to install. You should at this stage test the system on your local ethernet before installing an *mrouted* tunnel to the next upstream site. If you are on the Internet and wish to connect to the MBONE the mbone-oz mailing list will help you find a feed site.

- **Bandwidth** - Initial experiments should be restricted to your local ethernet with wide area links only attempted after you are more experienced.

For Further Information

To get more information on the multicast system or the tools used to implement you should read the Frequently Asked Questions (FAQ) and get hold of the documents mentioned in reference. In addition to this several mailing lists exist for the discussion of multicast topics:

aewg-mbone-request@nico.aarnet.edu.au
mbone-oz-request@internode.com.au
remconf-request@es.net

References

[1] Steve Casner.
"Frequently Asked Questions (FAQ) on the Multicast Backbone,"
6 May 1993,
ftp from **venera.isi.edu:mbone/faq.txt**

[2] Henning Schulzrinne and Steve Casner.
"RTP: A Transport Protocol for Real-Time Applications,"
IETF Draft, 20 October 1993,
ftp from **nic.ddn.mil:internet-drafts/draft-ietf-avt-rtp-04.ps**

[3] Mike Macedonia and Don Brutzman.
"MBONE, the Multicast BackbONE,"
January 24, 1994,
ftp from **taurus.cs.nps.navy.mil:pub/mbmg/mbone.hottopic.ps**

[4] Van Jacobson, Steve McCanne and Sally Floyd.
"A Conferencing Architecture for 'Light-weight Sessions'"
November 15, 1993,
ftp from **ftp.ee.lbl.gov**

[5] Van Jacobson, Steve McCanne and Sally Floyd.
"Lightweight Sessions - A new architecture for real-time applications and protocols"
November 30, 1993,
ftp from **ftp.ee.lbl.gov**

[6] Stephen Casner and Stephen Deering.
"First IETF Internet Audiocast"
July 1992,
ftp from **venera.isi.edu:pub/ietf-audiocast.article.ps**

[7] Douglas E. Comer.
"Internetworking with TCP/IP, volume 1"
1991
Prentice-Hall

[8] Stephen Deering.
"Host Extensions for IP Multicasting" RFC 1112,
August 1989,
ftp from **nic.ddn.mil:rfc/rfc1112.txt**

[9] Stephen Deering.
"MBONE - The Multicast Backbone",
March 3 1993,
ftp from **parcftp.xerox.com:pub/net-research/cerfnet-seminar-slides.ps**

[10] Steve Baker.
"Multicasting for Sound and Video"
February 1994,
Unix Review page 23-29

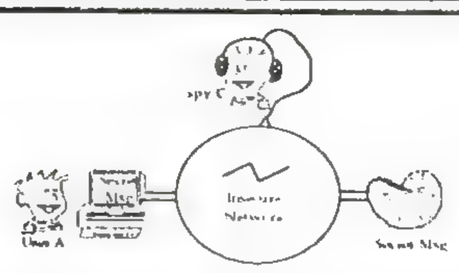
Authorisation and Privacy in a Networked World

Lawrie Brown
Australian Defence Force Academy

Introduction

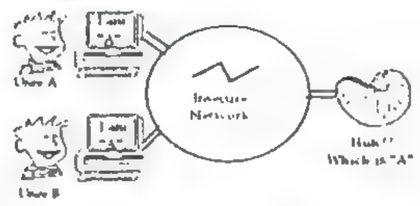
- traditionally networks were considered secure
- with spread of internetworking this is no longer true
- will discuss
  - why they are insecure
  - user authentication
  - privacy, particularly of email

Network (In)Security



- anyone with packet monitoring equipment can snoop on network traffic
  eg monitor logins and passwords on a local LAN

User Authentication



- need to validate identity of the user
  - local username and password
  - single-use passwords
    - paper-list
    - token generated
  - challenge-response
  - trusted authentication key servers

Challenge-Response

- basic technique used to ensure a password is never sent in the clear
- given a client and a server share a key
  - server sends a challenge vector
  - client encrypts it with private key and returns this
  - server verifies response with copy of private key
- in simplest form, keys are established before secure comminications is required
- in more complex forms, keys are stored in a central trusted key server

Challenge-Response

- this technique has been used as the basis for security extensions to
  - telnet
  - ftp
  - tcp/ip
- these are appropriate for use
  - in closed-user groups
  - with a small number of systems

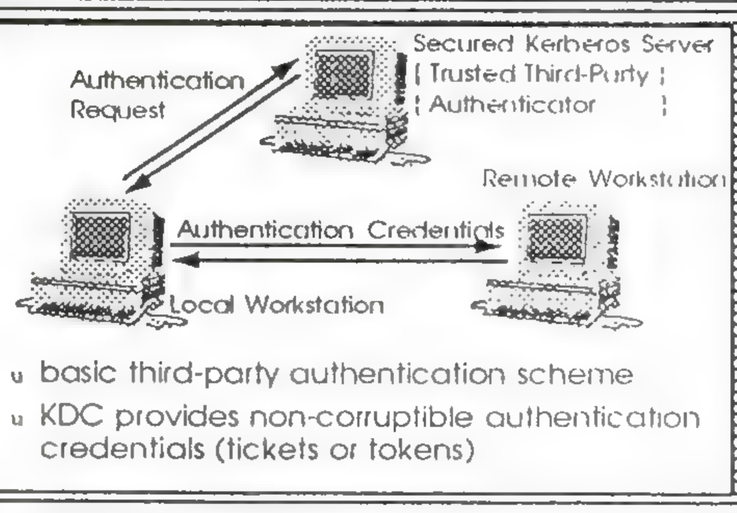
Authentication Key Servers

- have a key server trusted by all clients
- each client shares a key with the server
- the server can authenticate a client
- a client can request a ticket authenticating it to another client in order to obtain a service
- the client then sends this ticket to the other as proof of its identity and right to the service

Kerberos

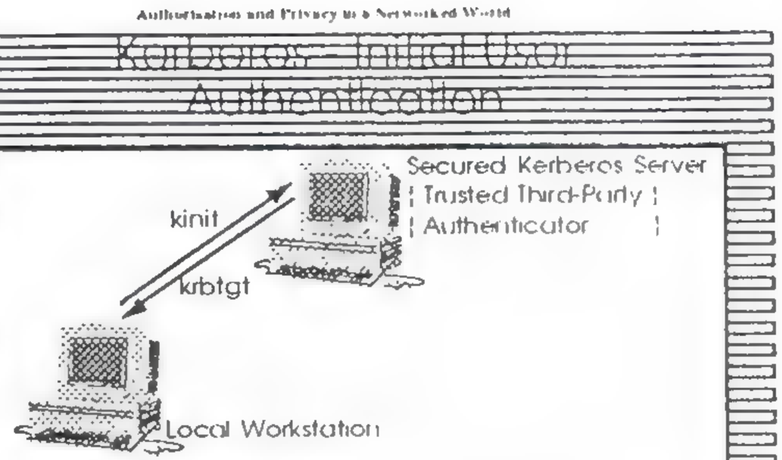
- trusted key server system developed by MIT
- provides centralised third-party authentication in a distributed network
- access control may be provided for
  - each computing resource
  - in either a local or remote network (realm)
- has a Key Distribution Centre (KDC)
- containing a database of
  - principles (customers and services)
  - encryption keys

Kerberos



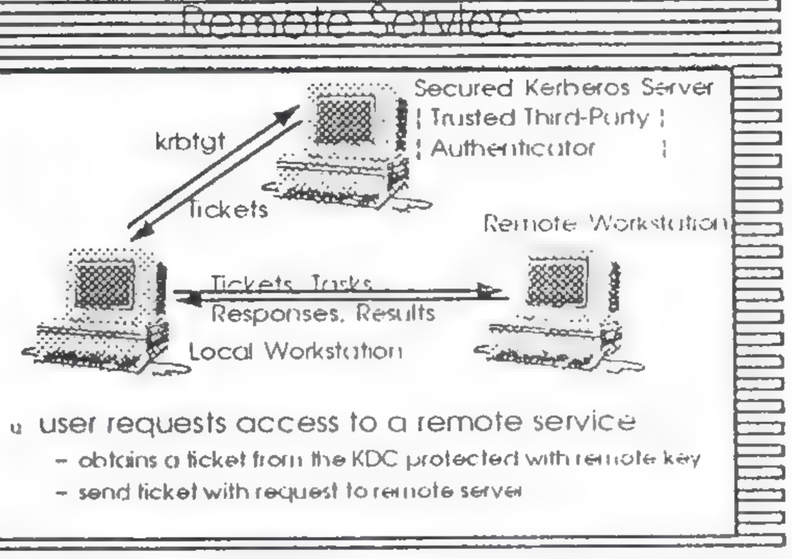
- basic third-party authentication scheme
- KDC provides non-corruptible authentication credentials (tickets or tokens)

Kerberos - Initial User Authentication



- user requests an initial ticket from KDC
- used as basis for all remote access requests

Kerberos - Request for a Remote Service

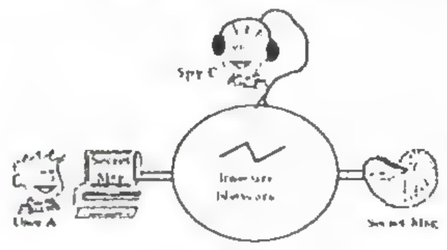


- user requests access to a remote service
  - obtains a ticket from the KDC protected with remote key
  - send ticket with request to remote server

Kerberos - In practise

- currently have two Kerberos versions
  - 4 : restricted to a single realm
  - 5 : allows inter-realm authentication, in beta test
- Kerberos v5 is an Internet standard
  - specified in RFC1510, and used by many utilities
- to use Kerberos
  - need to have a KDC on your network
  - need to have Kerbersed applications running on all participating systems
- major problem - US export restrictions
  - Kerberos cannot be directly distributed outside the US
  - crypto libraries must be reimplemented locally

Privacy



- need to protect the contents of messages by
  - physically preventing access to the communications link
  - cryptographically scrambling the data sent
- use a public or private key cipher
- need a key distribution scheme for keys used

Cryptographic Algorithms

- Private-key (symmetric) encryption
  - eg DES, FEAL, IDEA, LOKI
  - specifications for these ciphers are public
  - some have patent/licencing restrictions
- Public-key (asymmetric) encryption
  - eg RSA
  - RSA is patented in US/Canada by RSA Data Security Inc.
  - currently are only licencing use in US/Canada
  - legal status in rest of world is unclear
- Hashing Algorithms
  - produce a fixed size digest (summary) of a message
  - eg RSA MD2/MD4/MD5, HAVAL, SHS

IETF Common Authentication Technology (CAT)

- IETF identified a need for a standard way of invoking security services by applications
- formed the CAT working group to do this
- CAT provides an interface specification for accessing generic security services
- initially intends to use Kerberos v5
- is an Internet standard
  - specified in RFC1507, RFC1508, RFC1509, RFC1511
- not widely used yet, but will be in future

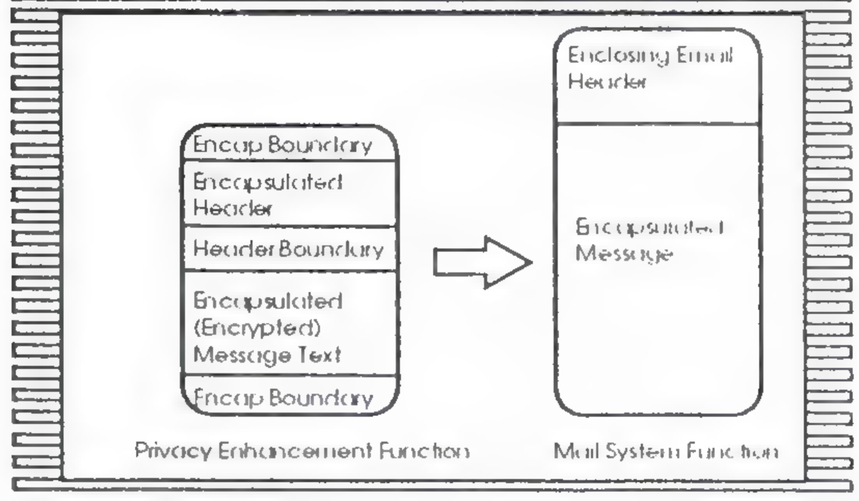
Secure Electronic Mail

- email is one of the most widely used and regarded network services
- currently message contents are not secure
  - may be inspected either in transit
  - or by suitably privileged users on destination system
- need a mechanism to provide
  - sender verification
  - message authentication
  - message privacy
- can't assume real-time access to a trusted key server

Email Privacy Enhancement Services

- confidentiality (protection from disclosure)
- authentication (of originator)
- message integrity (protection from modification)
- non-repudiation of origin (protection from denial by sender if public-key crypto used)

Email Encapsulation



PEM

- Privacy Enhanced Mail
- Internet standard for security enhancements to Internet (RFC822) email
  - developed by a Working group of the IETF
  - specified in RFC1421, RFC1422, RFC1423, RFC1424
- uses message encapsulation to add features
- confidentiality - DES encryption in CBC mode
- integrity - DES encrypted MIC (MD2/MD5)
- authentication - DES/RSA encrypted MIC
- non-repudiation - RSA encrypted MIC

PEM Key Management

- central key server (private-key)
  - requires access to on-line server
- public-key certificates
  - uses X.509 Directory Service Strong Authentication to protect key certificates
  - signed by a Certification Authority (CA)
  - CAs form a hierarchy to permit cross-validation of certificates
  - CAs must be licenced by RSA Data Inc
  - currently only licenced in US/Canada

PGP

- Pretty Good Privacy
- widely used de facto secure email standard
  - developed by Phil Zimmermann
  - available on Unix, PC, Macintosh and Amiga systems
  - free!!!!
- confidentiality - IDEA encryption
- integrity - RSA encrypted MIC (MD5)
- authentication & non-repudiation - RSA encrypted MIC
- uses grass-roots key distribution
  - trusted introducers used to validate keys
  - no certification authority hierarchy needed

PGP In Use

- all PGP functions are performed by a single program
- must be integrated into existing email/news
- each user has a keyring of known keys
  - containing their own public and private keys (protected by a password)
  - public keys given to you directly by a person
  - public keys signed by trusted introducers
- used to sign/encrypt your messages
- used to validate messages received

Sample PGP Message

```
---BEGIN PGP SIGNED MESSAGE---

This is a short message to illustrate
how a message is signed with PGP

Lawrie Brown

---BEGIN PGP SIGNATURE-----
Version: 2.3

iQBzAgUBLUh71ILpouh8ek7fAQGXHALsD3sf9Vo7I1krJdF6nKhwQ
8x5TdSFq9HCVf5qjmVXfdNRBL3otg4m1DhL0EoBbK9Vo/usgmGYz2
yffwL3hA2kRkBCu5f2KbMZ6/qSh6HQwR7YH+q5CKZQrtE5LbE1w==
=0gKD
---END PGP SIGNATURE---
```


PGP Problems

- possible problems
  - unlicenced use of RSA in US/Canada in contravention of patent (PD version of PGP is unlicenced)
  - there is a commercial licenced version in US/Canada
  - use outside the US is probably legal

Summary

- have discussed a range of privacy and authentication issues
- user authentication
  - closed user groups
  - Kerberos
- privacy
  - PEM
  - PGP

SLIDE 1 - PARALLELISATION

Opening Remarks.

My objective today is to explore the direction server technology has taken and where it will carry us in the foreseeable future.. I won't take too much time to detail the requirements for servers today, but a brief sojourn will help to paint the picture.

Very little current information is available on the topic of server parallelisation and what is available today or can be realistically expected.

I'll use the generic term 'server' to mean, UNIX RDBMS, I will use software to refer to server software and hardware to refer to server hardware. I won't be talking much of client or network architectures.

You may have noted that I work with Informix. The technical information I will use is derived from our current engine technology. It's called DSA - Dynamic Scalable Architecture. To hear more about DSA, you will have to risk calling me at the Informix office in Canberra. For the rest of my time here, I'll worry less about marketing terms, like 'DSA' and focus my attention on why parallelisation is a necessary and cost effective direction.

SLIDE 2 - GROUND RULES

Premises.

To get through the most important material I have to share today, I'll have to make a few unsubstantiated claims and work from these premises.

- **Server hardware has leap frogged server software.**

The rate of advancement in server platform hardware was exceeding the potential for server technology to exploit hardware advances. Hardware vendors have made a fork in the road. One direction - powerful uniprocessor technology. The other - multiprocessor technology. I would argue that the uniprocessor camp, will one day move to utilise those single processor developments in a multi processor architecture.

I'll explore why this is the expectation as I discuss the workings of a parallel server.

- **"Desktop oriented enterprise wide ..."**

Even a little exposure to the IT industry will bring an acquaintance with the 'the desktop revolution.' Some attribute this move away from centralised data processing solely to the IBM PC compatible. I choose to include the Macintosh and a plethora of alternative workstations in this picture.

It's also reasonable to include the notion of a client/server architecture and, more importantly the progressive loss of distinction between two elements of

the environment -the network and the database. In essence, communication and data management are becoming indivisible.

Slowly but surely, *data* is being distributed throughout the workplace. Even more gradually, *processing* of data is being distributed. I make this distinction between where data resides (and it may be replicated data) and where it is processed with a purpose.

The advent of client/server computing has not usurped the centralised data processing facility. It is still the *nature* of the data and how it is used, which dictates the storage method.

- **Data Holdings are undergoing geometric growth.**

The next slide alludes to some of what I have to say.

SLIDE 3 - DATABASE PERFORMANCE CRISIS

- **Σ Very Large Databases - VLDB.**

...the sum of which is known as VLDB.

People are finding more data to work with and more demanding work practices. The information super highway - why we can't have a less jingoistic label, I don't know. But the ISH (?) will bring the requirement for dealing with terabytes of data.

A terabyte is still an unconscionable mass of data - particularly if you've got a couple of terabytes and you're concerned about not losing them. I'll look at managing VLDB shortly.

A terabyte is 8 bits times a 1 with *twelve* zeroes behind it. Twelve zeroes? If that's hard to grasp, imagine the guy who had to fix some COBOL banking system to accommodate a monthly statement on the account for Bill Gates.

Alas, 'billion' and 'tera' have a lot in common - fortunately, the Americans take a somewhat deflated interpretation of billion to mean only nine zeroes, but Bill's working his way toward twelve zeroes anyway.

Enough of my envy - back to the point.

SLIDE 4 - DATABASES WILL FAIL

This is the underlying message of everything I have said. Is this bad news for Mr Informix and the other RDBMS vendors?

Given the evidence of insurmountable challenges, UNIX RDBMS vendors, venturing to trade in the Open Systems arena must deliver new solutions - not just a revamp of last years product with a lick of paint.

Necessity is the mother of invention.

SLIDE 5 - REQUIREMENTS

Why these issues?

There are many other salient issues in the IT industry today. I chose these three facets of the changing requirement for database management, because each represents a crucial element in the decision to use parallelisation.

1. Software technology must keep pace with hardware advances to deliver cost effective solutions based on optimal resource utilisation.
2. Servers must be consistent across a scalable hardware environment - ranging from standalone equipment to small corporate or divisional servers up to massive enterprise wide servers with features including distributed data management and so on.
3. Server architecture should be nonetheless platform independent. Software and hardware must be allowed to undergo structural change - for example, to accommodate VLDB applications. I'll discuss this last issue as I close on matters of directions and futures.

SLIDE 6 - SMP LC MP

Hardware Futures.

Hardware futures have been classified by three loose classes:

- existing **32** bit architectures;
- **64** bit; and
- what we term **90+** bit architectures.

What will come after 64 bit is still open to conjecture from those outside the hardware labs of the prominent vendors. There are arguments in favour of 90, 92 and 96 already on the table - the winning case is almost irrelevant; the succession of 64 bit platforms with machines furnishing still larger address space (whatever the intents and purposes) is unquestioned.

The evolutionary process is already well defined.

If the RDBMS vendors are to keep pace, a server architecture must be introduced to accommodate the structural or 'architectural' changes envisaged for main stream, Open Systems hardware vendors. These advances are already happening and are reaching the market from a plethora of specialist vendors.

SLIDE 7 - SMP WILL DOMINATE

The continuing trend, from even before 1993 is toward SMP machines. Existing technology is not invalid and I dare not suggest obsolescent, but the numbers tend to suggest that SMP has driven in the thin end of the wedge.

Slide 8 - What are the RDBMS Vendors Doing?

Five critical success factors exist. All this boils down to core parallelism in and Open Systems product, a UNIX RDBMS if you will.

Slide 9 - Porsche

Here's a nice picture of a sports car.

Slide 10 - Multithreaded, really?

Let's flare the ailerons for a slow, low altitude pass over the terrain of an RDBMS server with the features of core parallelism.

Technology in the field today is described as multithreaded. Let's look at the *granularity* of existing multithreaded engines - to borrow a term from the field of isolation levels and locking context. The multithreaded processing of such server engines is limited to an SQL process or task. To exploit hardware parallelisation, that is SMP architecture, the SQL process must be executed with far greater detail and finesse.

The two scenarios, OLTP and DSS, expose the weakness of an engine designed to allocate an SQL process to a CPU. The scenario on the right shows that resources (CPUs in this case, but also disks, I/O channels, tape drives and communications ports or sockets) can lie idle.

Slide 11 - Core Parallelisation

To continue the example, if the SQL process can be decomposed, each sub task can be dynamically allocated to any available resource. Of course this new level of arbitration brings new decisions and requires detailed management. The server must be aware of its environs - and readily advised of changes in circumstances - more on this later.

The scenario on the right, using CPUs are the limited resource, depicts an SQL process being dynamically broken down into sub tasks. Imagine this example applied to an SQL process further decomposed to exploit not only multiple CPUs but also multiple disks, partitions of memory and I/O bandwidth.

Slide 12 - Core Parallelisation

Where is the benefit perceived and what are the real world benefits of implementing core parallelisation?

SLIDE 13 - RECOGNISING HARDWARE LIMITATIONS

Here's the hierarchy:

- Processor
- Memory Cache
- Core Memory (RAM)
- Mass Storage (disk)

As I will explain, any engine task is subject to the most fundamental of design limitations in hardware. In short, we're all familiar with the notion that CPUs are faster than memory, which in turn is faster than I/O which in turn is faster than mass storage (disk) which in turn is commonly faster than communications.

Regardless of the units of measurement, it is accepted practice that, for efficient utilisation of all resources, each of these layers in the hierarchy must be dissociated from the layers above and below. Hardware manufacturers are countering these limitations in many ways, two of which are of immediate interest:

- caching, the idea of 'drafting' active data up through the levels of hierarchy to be temporarily available at the layer where it is being actively manipulated; and
- parallelisation, allowing multiple channels of connection between the layers.

There's a lot of attendant technology, light weight process architecture in operating system design and so on, but I'll focus on the design issues which fundamentally effect database behaviour and resultant performance.

SLIDE 14 - PARALLEL TASKS

This graph is to be read vertically, imagining an SQL process executing through time, starting at the base of the graph and progressing upwards. The serial notion on the left is what we are putting up with today. The middle ground, shows the reduction in overall execution of the SQL process if

One minor point, you may be tempted to think that a read or a write is an SQL process - not so. The four steps here are involved in almost all SQL processing. Even a seemingly 'read only' task may involve a write to a lock and will almost certainly involve a write to a log file in some manner.

Finally, the third approach, on the right, is a depiction of the fine granularity of a core parallelisation of an SQL process. Now, how can this be achieved?

SLIDE 15 - PARTITIONING

We are probably all familiar with the notion of partitioning, the business of striping disks has been around and in use for a long time. The big question is simply how to organise database processing to allow the perfect situation where the large disk access task - as depicted on the left - can be restructured to execute in parallel - as depicted on right.

I am convinced that the server must also partition CPUs, memory, and all forms of resource management, I/O and so on. Furthermore, the engine must be able to respond dynamically to changing requirements. In practice, tuning the server for OLTP will disadvantage DSS applications and certainly impact the processing of batch operations.

SLIDE 16 - VIRTUAL PROCESSORS

To describe in real terms, what can be achieved with core parallelism, I will have to lean on some Informix slides to examine how it's done.

This diagram shows two concepts, *'fan in'* and *'fan out.'* A variable number of user sessions, composed of SQL processes is being decomposed into sub tasks. As sub tasks are processed, I/O requests, including typical reads and writes are being *'fanned out'* to optimise parallel channelling to CPUs, disks and so on.

These sub tasks are being 'picked up' by Virtual Processors, or VPs. Note that there is no 'governing' VP, no single point of arbitration.

Virtual Processors are basically generalised database-level processes that are able to multiplex to any number of requests, that is any number of user sessions. VPs are based on the principles of multithreading with the database context.

This scenarios does not result in a plethora of VPs. There are three classes of VPs.

- **asynchronous I/O -** handling I/O channels
- **CPU -** executing the lion's share of all processing, that's the largest population group in the VP pool; and
- **connection -** for managing user sessions over any transport, network or character/serial terminals.

Back to my point on no single point, no bottleneck per se. The CPU class VP is also being allocated to the engine processes. Not only user sessions are entering the pool. Engine processes, like the query optimiser, require cycles. As the head count of connected users grows, the connection class VPs doesn't grow in proportion. As the number of I/O requests begins to 'queue up', the number of I/O VPs may not necessarily increase, unless there's a good reason. I give you some scenarios shortly.

The query optimiser is reading configuration information, usually held in unshared memory, to determine the best process plan for executing the high

level SQL statement, decomposing the sessions into sub tasks and introducing parallel processing. By having details of how many physical CPUs, disks etc exist, and what arbitrary allocation rules have been imposed, the query optimiser can make 'load balancing' adjustments to distribute I/O, for example, across available resources. Of course, more detailed information must exist; where the data resides, what mirroring is in effect, maybe some resources are down etc.

SLIDE 17 - RESOURCE BLOCKING

Here's the single most convoluted and difficult concept in the scheme.

Keep in mind that the pool of VPs can be queued against resources. Let's just work with CPUs for the purposes of explanation. The normal overheads of context switching, by CPUs (administered by the host operating system) carries a lot of 'baggage' for application processes, like the database engine. Instead, with the sub tasks being queued to VPs, each VP has a minimal context, only a small stack, heap space and instruction cache. When a physical CPU is swapped (under the administration of the database engine) from one VP to the next, even if it's a different class of VP, the context overhead is greatly reduced. This make it efficient to swap out VP that has a resource block.

A resource block is any kind of delay in resource availability. If the VP thread is waiting for a lock, it can swap out. Existing 'multithreaded' architectures would be blocked by this resource being unavailable. Even under ideal conditions, the engine thread would be swapped by the operating system, involving much greater context overheads.

Here's an example that constitute most of the resource contention and blocking problems which plague existing products in the market.

"Hot Spots" occur where the engine has a limited capacity to process tasks resulting from SQL processing. Having a single point of process arbitration is one easy example, the idea of a 'governor.' More insidious is the I/O processes created by logging transactions. Whenever an engine has to process a large number of transactions being committed at the same time - common to batch, but also associated with DSS tools with update facilities, the queue for writing logs can bring all other engine processing to a stand still.

This problem of hot spots requires resource governing from the engine and control by management tools for the DBA to oversee the tuning. I'll discuss manageability in due course.

Efficient processing is analogous to efficient business in any sphere.

As a rule, "Don't send too many messages; make each message meaningful and efficient." This translates into, "Don't have too many meetings." This also means, "Ensure the meeting accomplishes something, not just to agree on the time and place for another meeting."

In VP terms, this is becoming clear; When a VP is allocated a physical CPU there must be work to process without delay and, hopefully completed without needing to generate another queue element.

Well, that's a lot of information supporting just one slide, but, consistent with the notion of fine granularity, the efficiency of a parallel server manifests itself at a low level.

Slide 18 - Virtual Processors

One of the critical success factors identified is high availability. One of the failings of existing servers is just this lack of high availability. Fault tolerant servers are peddled by many of the players. Avoiding the unplanned down time is possible. Planned down time is equally unacceptable.

Slide 19 - Configuration

The rather theoretical sliding bar in this slide is an easy way to depict the 'balancing act' to tune the engine performance. This is a critical factor in parallelism. All the server load is being handled by one 'instance.' If we instantiated the engine for each new application or for different users or whatever the motivation, we would be throwing away the advantages of light processes, the VPs and the context switching advantages.

Having one instance keeps the engine and optimiser in touch with all task queues and resultant loads. Having one instance also puts the operating system as far out of the picture as possible. If we had more than one instance, we would still be relying upon the operating system to do something it's not designed to handle: swapping consecutive database threads.

Having a pool of VPs is a flexible environment. Rules for adding and removing VPs can be applied while the engine continues processing user sessions. Moving from the days' OLTP to a schedule evening of batch can be automated - More I/O VPs, less core processing and more physical CPUs to batch processing in parallel, thereby 'feeding' the parallel access of disks, for example. The proof of the pudding here is to measure the realised I/O throughput of the mass storage under engine load.

If a disk is capable of 10 megabits per second maximum throughput, when the engine can exert this load in a sustained transaction processing situation (like OLTP) we've realised the cost benefit figure that may have encouraged us to buy the hardware in the first place. Parallel I/O processing, using I/O VPs and enough physical CPUs can create this load. The effect is to process all engine sessions as quickly as possible on the given platform.

Within the constraints of maintaining 24x7 availability, the server must be configured 'on the fly.' As we have seen, the configuration process will be necessary to 'advise' the engine and the query optimiser, of the changing characteristics of the planned work load.

Slide 20 - Conclusion

Core level parallelisation of the server architecture is a giant step in performance. It's clear cut theory which has been applied with proven success by the hardware vendors. The lag in delivery of software vendors was an issue. More detailed information is available from copyright material produced by independent third parties, I can make this available to anyone who wishes to contact me.

Parallelism in RDBMS Servers

DSA - Dynamic Scalable Architecture

INFORMIX

Premises

- Server hardware has leap frogged software
- "Desktop oriented enterprise wide ..."
- Data holdings are showing geometric growth
  - Σ Very Large Databases - VLDB

INFORMIX

Database Performance Crisis

- Dramatic growth in database size
- Ever increasing performance needs
- New generation of multiprocessor systems

INFORMIX

Existing database architectures will fail.

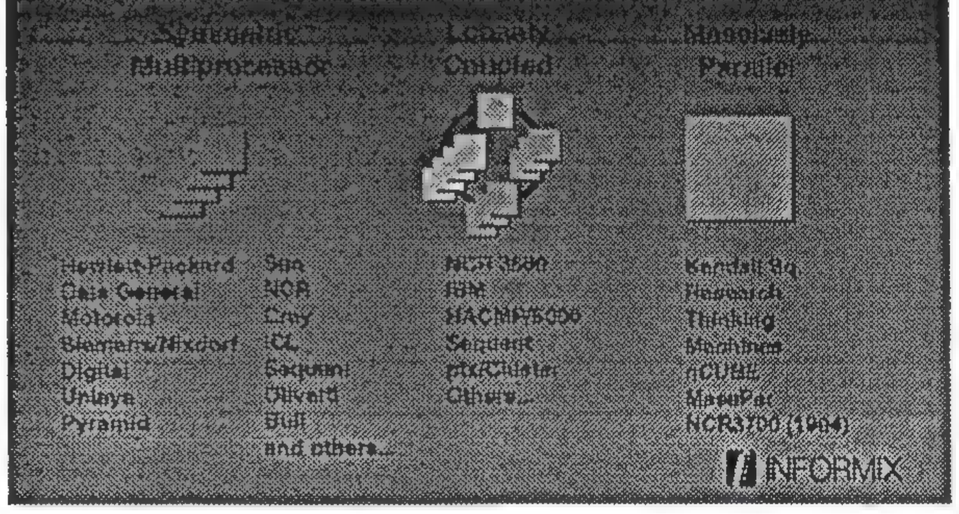
INFORMIX

Next-Generation Requirements

- Open systems implementation
- VLDB applications
- OLTP, DSS, EIS and batch
- 24x7 Availability, DBA facilities
- Application transparency
- Hardware indepedence
- Users and Queries en masse

INFORMIX

High-Performance Hardware



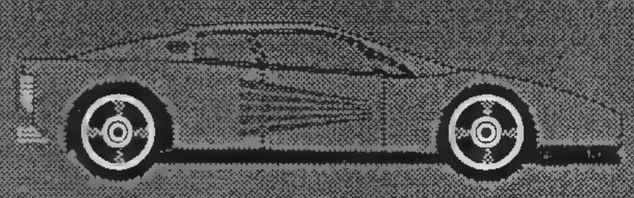
INFORMIX

Critical Success Factors

- Hardware architecture & platform independence
- Transparent to applications
- Core internal parallelism
- Efficient processing
- Maximum on-line manageability

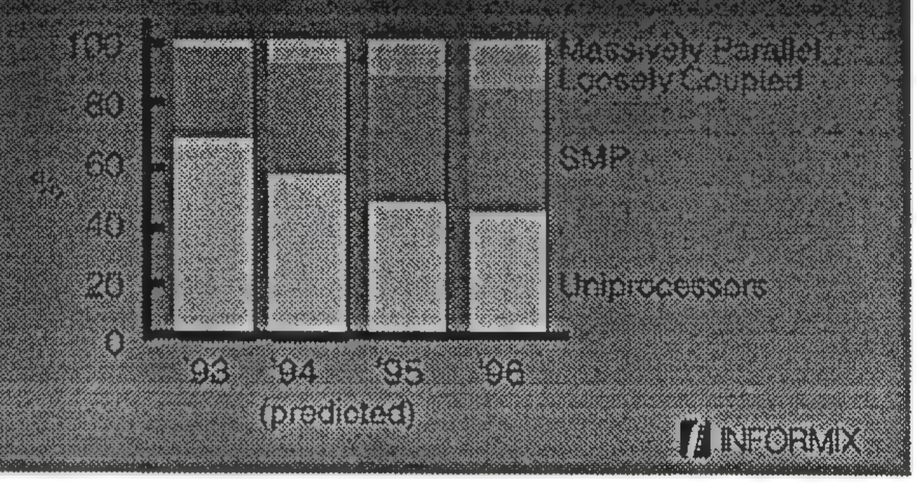
INFORMIX

Red Sport Car



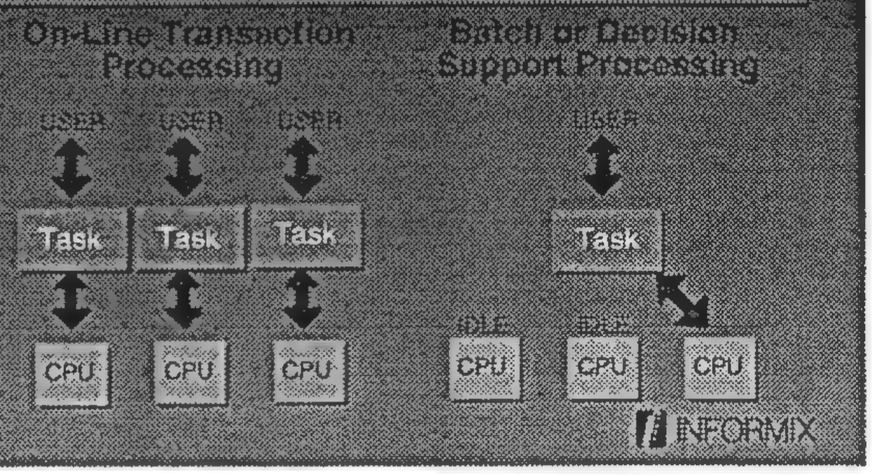
INFORMIX

SMP Will Dominate

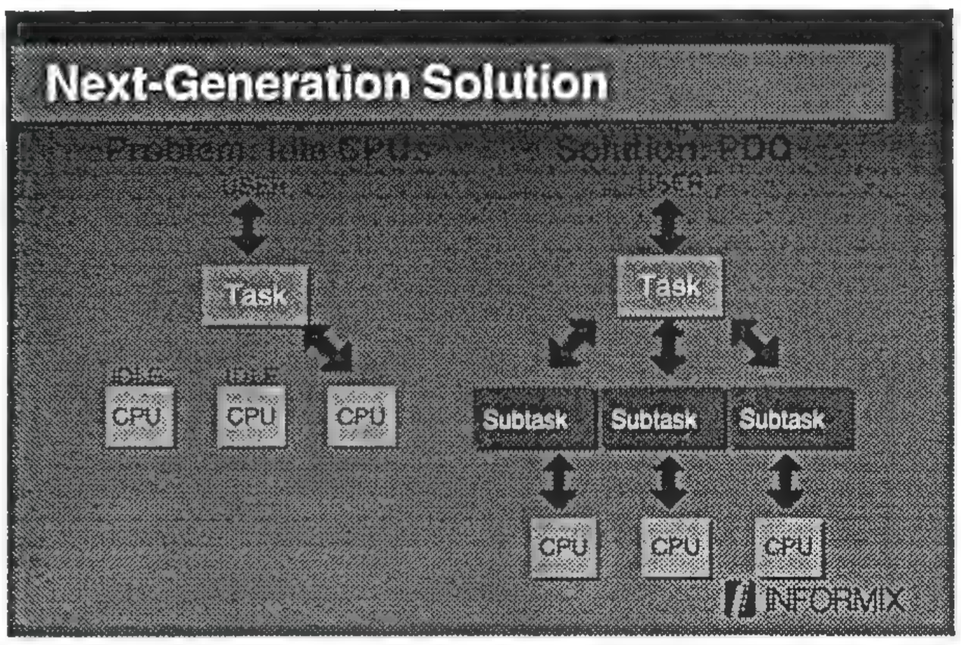


INFORMIX

Limitations of Current RDBMS Parallelism



Next-Generation Solution



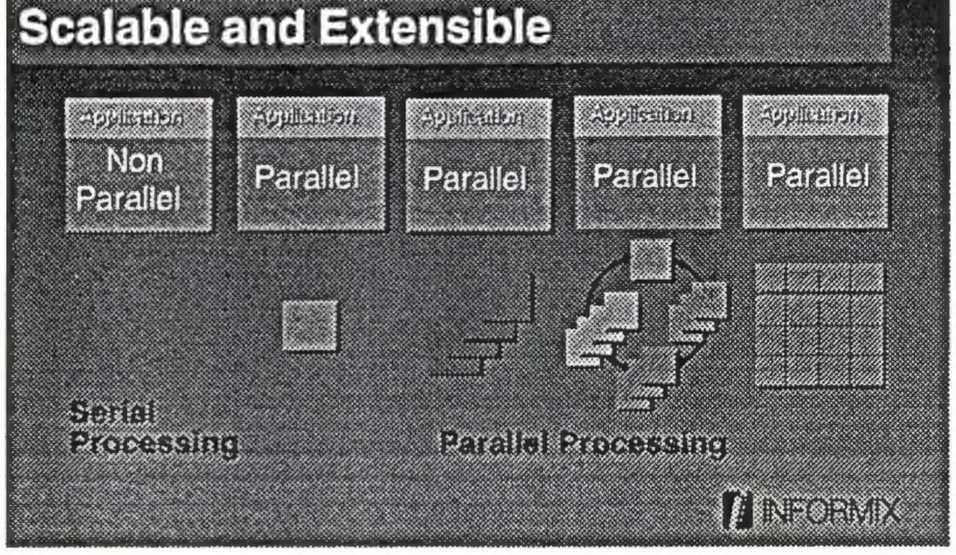
Where Parallel Processing Can Make the Biggest Difference

- Batch jobs, reports
- Index builds
- Bulk loads/unloads
- Backup/restore
- Altering/reorganizing tables
- Mass updates/deletes

INFORMIX

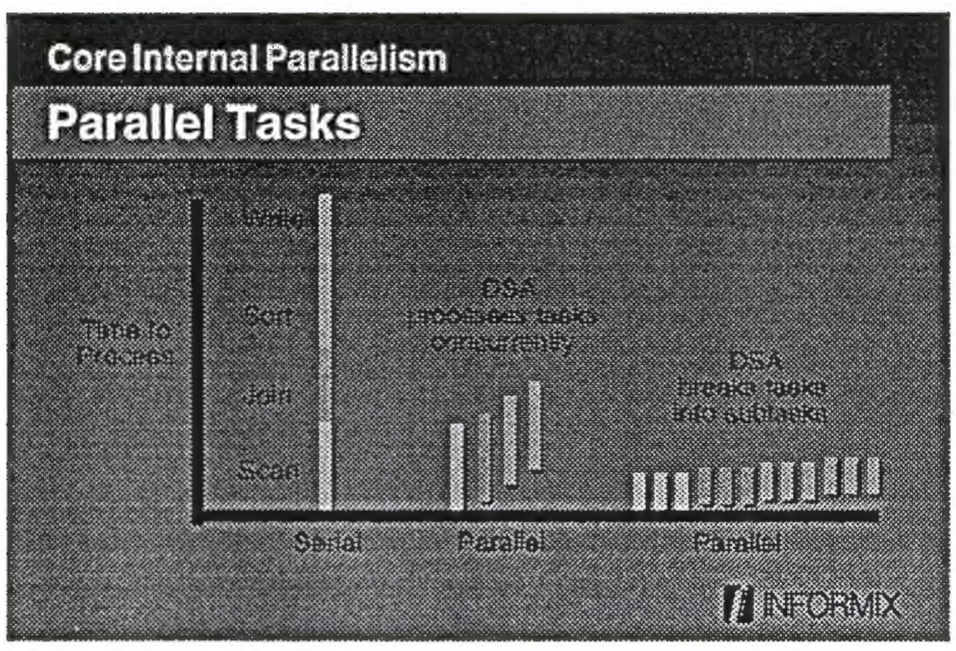
Transparent to Applications

Scalable and Extensible



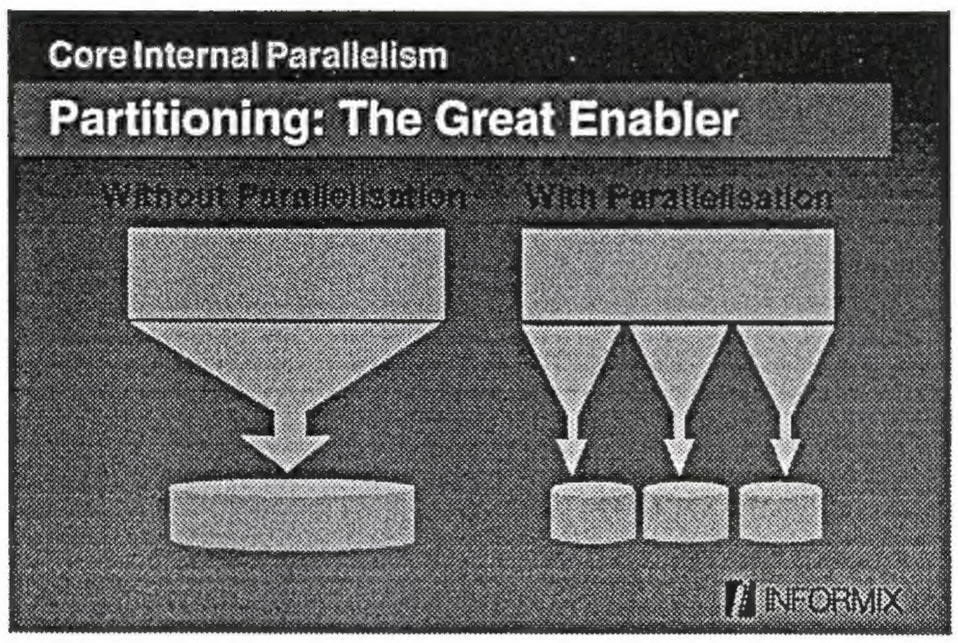
Core Internal Parallelism

Parallel Tasks



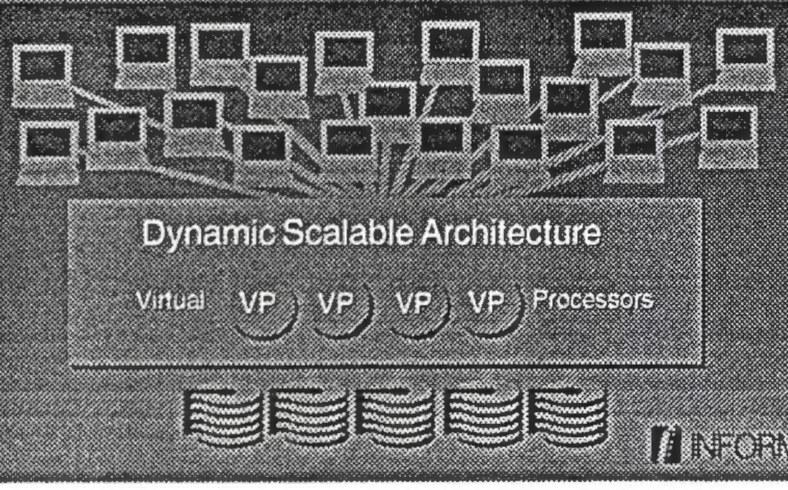
Core Internal Parallelism

Partitioning: The Great Enabler



Core Internal Parallelism

Parallel Tasks



Core Internal Parallelism

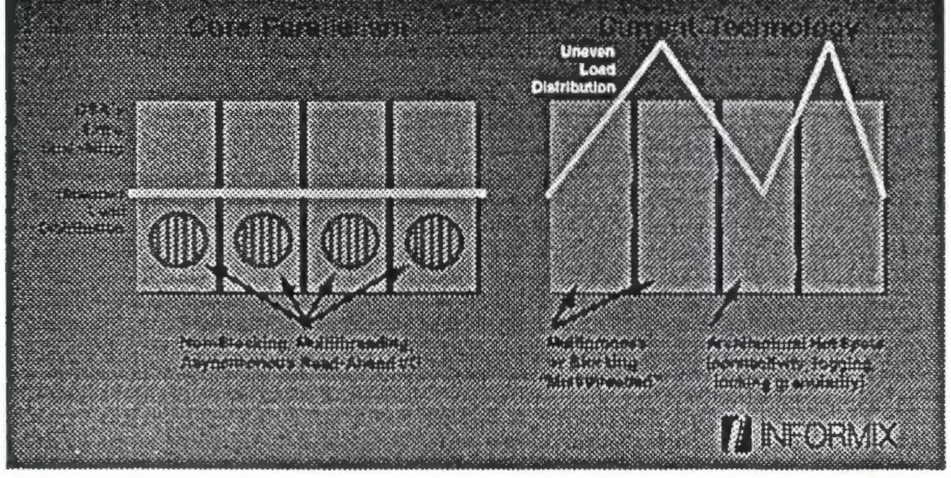
Advantages of Partitioning

- Distributes I/O across disks for parallel scans
- Unavailable partitions are skipped to maintain high availability
- Uninvolved partitions are skipped to increase performance
- Enables backup and restore of single partitions
- Transparent to applications and users
- Distributed partitioning addresses loosely coupled architectures

INFORMIX

Efficient Processing

Multithreading



Maximum On-Line Manageability

Mission-Critical Functions

On-Line
Dynamic Reconfiguration
- Virtual processors
  - Client connections
  - Server pool
- Degree of parallelism
- Memory management

On-Line
Backup/Recovery
- Parallel
- Table
- Partition
- Incremental

INFORMIX

Maximum On-Line Manageability

Concurrent Processing